Clinic Appointment Booking System: Streamlining Patient Intake

COMM 394 - Coding Literacy for Managers

ASSIGNMENT OVERVIEW

In this assignment, you will build a web-based appointment booking system that transforms how clinic reception staff handle phone bookings. The goal is a professional, working tool that helps staff book faster, reduce errors, and capture complete information on the first call.

WHY THIS MATTERS FOR MANAGERS

This project teaches you how to scope and validate a real business problem before you code, and how to build tools that actually fit how work happens. Many failed software projects start with technology and look for problems later. Here, you start with a genuine intake pain point in a clinic, use design thinking to understand the people and pressures involved, and then build a tool that solves that specific problem. By the end, you will understand why business strategy and user needs drive technical decisions, and how to check whether a tool actually works for the people who use it every day.

Your app supports staff to:

- Enter appointment details in a clear, guided form
- Prevent double-booking through conflict checks
- See today's appointments in a dashboard that supports quick decisions

The system addresses common clinic problems:

- Manual processes slow calls while patients wait on the phone
- Incomplete or inaccurate details lead to callbacks and frustration
- Long calls cause call queues and missed calls

Your goal is a user-friendly system that fits real phone work in an Ontario clinic and supports accurate, efficient appointment booking.

ASSIGNMENT GUIDELINES

Imagine you operate a small independent clinic in Ontario that uses a manual phone-only booking system. Staff write on paper or type into a basic spreadsheet while managing calls.

Clinic Details

- **Clinic name and type:** Example: Maple Family Health (family practice), North Shore Urgent Care, Valley Medical Centre (specialty)

- **History and current situation:** When the clinic opened, how it has grown, and how appointments are managed today (paper, spreadsheet, or both)
- **Location and size:** Ontario town or city, number of reception staff, number of doctors, number of exam rooms
- **The appointment problem:** Typical call length, common errors, dropped calls, or callbacks due to missing or wrong information

Market Context

Explain why clinics like yours need better appointment booking solutions. Connect to issues such as:

- **Staff frustration:** Time spent searching paper files or spreadsheets, repeating information, and fixing errors
- **Booking delays:** Calls that take eight to ten minutes with callers waiting on hold
- **Data errors:** Misspelled names, wrong contact numbers, incorrect doctor or time, and missing information
- **Call queue backup:** Missed calls and long wait times when staff are tied up on long booking calls

Technical Standards

This assignment uses vanilla HTML, CSS, and JavaScript with no frameworks or libraries. You will hand-code all HTML structure, write your own CSS for styling, and implement all logic in plain JavaScript. This approach teaches you the fundamentals of how web applications work at the core level.

PART 1: DESIGN THINKING CANVAS (25% OF GRADE)

You create a Design Thinking Canvas that explains the intake problem, the people affected, and how your proposed web app addresses it.

Format

- PowerPoint or PDF
- Eight to twelve slides
- Organised into four sections

***Note**: You can refine your problem as you go. Iteration is expected.*

SECTION 1: CONTEXT (1-4 SLIDES)

Purpose

Set the scene for your clinic and show why phone intake creates operational challenges.

Slide 1: Clinic Profile

Include:

- Clinic name and type (family practice, urgent care, specialty clinic)
- Location (Ontario town or city)
- Size (reception staff, doctors, exam rooms)
- Booking model (phone-only, no online booking, no walk-ins)
- Call volume (estimated daily calls, such as 40-60 per day)
- Peak times (for example 8:30-9:30 AM and 4:00-5:00 PM)
- Patient demographics (age ranges, catchment area, language mix if relevant)

Slide 2: Current Phone Intake Operations

Describe how booking works today:

- **Process:** How staff answer calls, gather information, and record appointments
- **Average call duration:** Typical time per appointment call
- **Data captured:** Examples include name, contact, doctor, date and time, reason for visit
- **Call management challenges:** Dropped calls, long hold times, calls that go to voicemail
- **Accuracy problems:** Common errors such as wrong doctor, wrong date or time, misspelled names, wrong phone numbers, duplicate bookings
- **Information gaps:** Important information that should be collected but often is not, such as allergies, Patient ID, or relevant medical notes

Slide 3: Stakeholder Needs

Identify the key people involved and what each person needs from an efficient booking system:

- **Reception staff:** Faster call handling, clear guidance on required fields, confidence in data accuracy
- **Office manager:** Smoother call flow, fewer callbacks, predictable scheduling, fewer staff frustrations
- **Clinic owner:** Lower operational costs, higher patient satisfaction, reduced missed appointments, staff retention
- **Doctors:** Complete and accurate information before the visit, clear allergies and special notes, fewer booking errors
- **Patients:** Quick and accurate booking, short call duration, correct appointment details, trust that their information is captured

Slide 4: Pressures And Constraints

Identify the specific time, accuracy, and operational pressures that make the current system difficult:

- **Reception staff pressures:** Time pressure (calls queued, patients waiting on hold), stress from repeated questions or corrections, accuracy risk under time pressure
- **Call management constraints:** Limited ability to search records quickly, no structured data entry to guide completeness, no real-time conflict detection

- **System constraints:** Manual or spreadsheet-based tools lack validation, no quick feedback on errors or conflicts, no visual summary of daily workload
- **Financial and patient impact:** Staff hours spent on corrections and callbacks, patient frustration leading to complaints or missed appointments, reduced clinic efficiency

SECTION 2: PROBLEM DEFINITION (1-3 SLIDES)

Purpose

Define one clear, specific phone intake problem that your web app solves.

Primary Problem Statement

Use one precise sentence with WHO, WHAT, and the CONSTRAINT.

Template:

"[Staff role] at [Clinic Name] cannot [phone intake task] efficiently because [constraint], which leads to [specific negative impact]."

Examples of constraints:

- Manual search of records
- No quick way to verify patient details
- No structure for required information
- No way to see conflicts during the call

Supporting Evidence

Select at least two or three types that match your problem:

- **Time measurements:** Average call time, time spent searching records, time managing hold queues
- **Error rates:** Percentage of incorrect or incomplete entries, duplicate bookings, or wrong doctor or time
- **Operational losses:** Staff hours spent on corrections, callbacks, and follow-up confirmation
- **Staff feedback:** Direct quotes about frustration with the current call process
- **Patient complaints:** Comments about long wait times, wrong appointment details, or repeated questions

Root Cause Analysis: 5-Why Method

Use the 5-Why method to move from surface symptoms to a clear root cause. Show each step.

Example pattern:

Problem: Appointment booking takes eight to ten minutes per call.

Why: Staff spend time searching for existing patient records while the caller waits.

Why: They do not have fast Patient ID search in one place.

Why: Current tools are paper-based or simple spreadsheets with weak search.

Why: The clinic never invested in a structured digital intake system.

Root cause: No rapid, structured digital intake and search system exists, so staff lose time on manual steps.

Impact Quantification

Include two or three problem-specific metrics, such as:

- **Call time reduction:** For example, from eight to ten minutes down to four to five minutes
- **Accuracy improvement:** For example, lower error rate from fifteen percent to two percent
- **Queue improvements:** Fewer dropped calls each day
- **Time savings:** Staff hours saved each month
- **Completeness:** Higher share of appointments with all required data captured on the first call

SECTION 3: SOLUTION OVERVIEW (1-3 SLIDES)

Purpose

Show the improved phone intake experience after your web app is in place.

Required: Phone Call Scenario Walkthrough

Present a realistic before-and-after scenario.

Include:

- **Specific staff name and time:** For example, "At 8:45 AM, Sarah answers a call from a returning patient."
- **Realistic script highlights:** What Sarah asks, what the patient says, and what Sarah enters into the system
- **Screen references:** Mention which screens or features the staff member uses during the call
- **Time savings:** Clear comparison between old and new call duration
- **Accuracy and completeness:** Show that all required fields are captured and checked
- **Measurable outcome:** Connect back to your problem metrics to show that the problem is addressed

SECTION 4: FUNCTIONAL REQUIREMENTS (CANVAS) (1-2 SLIDES)

Purpose

Describe the custom features your web app introduces to tackle the specific problem from Section 2.

For Each Planned Custom Feature, Describe:

- **Feature name:** Clear, short name that staff could understand
- **How it addresses your problem:** Explicit link back to your primary problem statement
- **Description:** What the feature does, when it is used in the call, and how staff interact with it
- **Expected impact measurement:** Estimated effect on call time (seconds saved), accuracy (percent improvement), completeness (fields auto-filled), or queue performance (seconds per booking)

You do not need to state where each feature sits in the code at this stage, but the features you describe here must match what you deliver in the web app.

CUSTOM FEATURES: EXAMPLES

You must design and implement at least three custom features that directly solve your Canvas problem. Here are concrete ideas to spark your thinking, organised by complexity. These examples are referenced again in the Custom Features Requirements section of Part 2.

Easy Features

- **Quick-reference allergy alert:** In the dashboard, highlight any appointment where allergies are present (not "None known") with a colour or icon. This helps staff spot high-risk bookings at a glance.
    - **Expected impact:** reduce missed allergy alerts to near zero.
- **Appointment priority flag:** Add a priority dropdown to the booking form (routine, urgent, same-day sick visit). Display the priority as a visual indicator in the dashboard using colour or icons. This helps staff and doctors identify time-sensitive appointments.
    - **Expected impact:** improve appointment triage accuracy by 25-40 percent.
- **Call timer:** Display a running timer that starts when the booking form is opened and stops when the appointment is saved. Show a visual warning (colour change) if the call exceeds a baseline such as five minutes. Display the final call duration in a confirmation message.
    - **Expected impact:** reduce average call time by 30-40 percent.

Medium Features

- **Appointment confirmation script:** Add a button that generates a formatted confirmation statement staff can read back to the patient (for example, "I have you booked with Dr. Patel on Tuesday, March 15th at 2:00 PM for a follow-up visit. You noted an allergy to penicillin. Is that correct?"). This ensures accuracy before saving.

    - **Expected impact:** reduce booking errors by 60-80 percent.

- **Same-day appointment alert:** When staff enter a Patient ID and select an appointment date, check if that patient already has another appointment on the same day. If so, display a warning with the existing appointment details. This prevents accidental double-booking of the same patient.
    - **Expected impact:** reduce same-patient duplicate bookings by 80-90 percent.
- **Doctor appointment count:** Display a panel showing how many appointments each doctor has for the currently selected appointment date. Update the count dynamically as the date selection changes. This helps staff identify which doctors have lighter schedules.
    - **Expected impact:** reduce call time by 1-2 minutes per booking.
- **Edit appointment:** Add the ability to edit an existing appointment by Appointment ID. Provide a text input where staff enter an Appointment ID and a Search button that retrieves the matching appointment. If found, display the current appointment details and allow staff to select which field to update (for example, Doctor, Appointment Date, Appointment Time, Reason for Visit, Allergies, Special Notes, Contact Number, Email Address). After selecting a field, show the current value and provide an input to enter the new value. The system must re-validate the changed field and re-check for scheduling conflicts if Doctor, Appointment Date, or Appointment Time is changed. If a conflict exists with another appointment, show an error and block the save. After successful update, refresh the dashboard immediately.
    - **Expected impact:** reduce time spent correcting bookings by 60-70 percent.

Hard Features

- **Returning patient lookup:** When staff enter a Patient ID, automatically search the appointments array and display that patient's appointment history, including previous visit dates, doctors seen, and reasons for visit. This speeds up booking for returning patients and supports continuity of care.
    - **Expected impact:** reduce call time for returning patients by 2-3 minutes.
- **Recurring appointment series:** Add an option to book a series of recurring appointments (for example, every two weeks for six visits). The system should generate all appointments in the series, check each for conflicts, and report any that could not be booked. This supports patients needing ongoing care such as physiotherapy or chronic disease management.
    - **Expected impact:** reduce booking time for recurring patients by 70-80 percent.
- **Patient database:** Create a separate patient registry that stores patient details (Patient ID, name, date of birth, contact number, email, allergies) in Local Storage. When staff enter a Patient ID in the booking form, automatically retrieve and populate the patient's information. Include the ability to add new patients to the registry and update existing patient details. This eliminates re-entry of patient information for returning patients and ensures consistent data across appointments.

  - **Expected impact:** reduce call time for returning patients by 2-3 minutes and reduce data entry errors by 50-60 percent.

These are examples. You may use any of these, modify them, or invent your own, as long as each feature solves a concrete problem from your Canvas and improves one of these dimensions: call speed, accuracy, completeness, queue flow, or staff confidence, with an estimated measurable impact.

Grading Notes

To achieve the highest marks, you need to deliver at least one Hard feature, at least one Medium feature, and at least one Easy feature. This combination demonstrates range across technical complexity and business value.

If you are unable to complete a Hard feature, you can substitute two Medium features. If you are unable to complete a Medium feature, you can substitute two Easy features. These substitutions allow flexibility while maintaining the expected depth of work.

For example, valid combinations include:

- One Hard, one Medium, one Easy (target)
- One Hard, three Easy (substituting two Easy for the Medium)
- Three Medium, one Easy (substituting two Medium for the Hard)
- Two Medium, three Easy (substituting two Medium for the Hard, and two Easy for one Medium)

Remember that technical complexity is only part of the assessment. Business value matters equally. A feature that directly solves your Canvas problem and delivers measurable improvement to call speed, accuracy, or staff workflow will score higher than a technically impressive feature with weak business justification. Quality of implementation and clear alignment with your problem statement matter more than quantity. A well-executed Easy feature that solves a real intake problem scores higher than a poorly implemented Hard feature with unclear business benefit.

PART 2: WEB APPLICATION (75% OF GRADE)

You implement a single-page or simple multi-page web app that matches your Canvas and supports real phone intake work.

Data Validation

All user input must be validated before saving an appointment. If validation fails, the app blocks the save and shows a clear message with alert().

Validation Requirements

Field	**Validation**	**Error Message**
Patient Name	Not empty, at least 2 characters	"Patient name is required (minimum 2 characters)"
Date of Birth	HTML date input	Browser native message

Patient ID	Exactly 6 digits	"Patient ID must be 6 digits (for example 123456)"
Contact Number	Numeric, at least 10 digits	"Contact number must be 10 or more digits (for example 6135551234)"
Email Address	Contains @	"Please enter a valid email (for example john@example.com)"
Doctor Selection	Not empty	"Please select a doctor"
Appointment Date	HTML date input, no past dates	Browser native message
Appointment Time	HTML time input	Browser native message
Reason for Visit	Not empty	"Please provide a reason for visit"
Allergies	Required. Either allergies or "None known"	"Allergies field is required. List allergies or enter "None known""

Implementation Approach

- Use HTML native inputs for dates and times
- Use JavaScript validation for Patient ID, contact number, and email
- Block form submission until all checks pass
- Use alert() for clear, specific feedback on errors
- Follow the exact wording for error messages in the table above

Appointment Date Window

Appointments may be scheduled from the current date forward without a specific end limit. However, your system should handle any future date that staff select. You do not need to enforce a maximum date in the future (for example, staff may book appointments six months ahead if needed). The key validation is: no past dates.

Visual Design

The interface should look like a real healthcare staff tool.

Standards:

- Use calm, professional colours such as blues, greens, and greys
- Use clear fonts such as Arial, Helvetica, or other sans-serif fonts
- Order fields to match the flow of a live intake call
- Label every input clearly
- Use spacing and grouping so staff can scan quickly
- Show appointments in a table with clear headings and borders
- Make buttons easy to see and click (Submit, Search, Clear)

- Avoid visual clutter and decoration that draws attention away from the task

COMPONENT 1: PHONE INTAKE FORM

Mandatory Requirements

Your main booking form must include the following fields and actions in this order:

1. Patient Name (text, required)
2. Date of Birth (date input, required)
3. Patient ID (text, required, 6 digits)
4. Contact Number (text, required, 10 or more digits)
5. Email Address (text, required, contains @)
6. Doctor Selection (dropdown, at least three doctors, required)
7. Appointment Date (date input, required, no past dates)
8. Appointment Time (time input, required)
9. Reason for Visit (text input or dropdown, required, examples such as Annual Checkup, Follow-Up, New Patient, Urgent Care)
10. Allergies (text input, required, list allergies or enter "None known")
11. Special Notes (text area, optional, maximum 200 characters)
12. Submit button (saves appointment if all validations pass)
13. Clear button (resets form for the next call)

The form must:

- Enforce all validation rules listed in the Data Validation section
- Follow the professional visual design standards
- Support a smooth flow for a live phone call

You may place one or more of your custom features in this form, as long as they connect to your Canvas problem.

COMPONENT 2: DATA MANAGEMENT

Mandatory Requirements

Your app must:

- Define an Appointment object constructor that stores all required data fields
- Maintain an array of appointments in memory (Example: appointments = [])
- Save all appointments to Local Storage after each change
- Load appointments from Local Storage when the page loads
- Delete appointments from the array and save changes to Local Storage
- Prevent duplicate bookings using a rule: same Patient ID plus same doctor plus same date and time
- Enforce that the allergies field is never empty in saved appointments

Each Appointment Object Must Store

- Appointment ID (auto-generated, sequential, such as APT001, APT002, and so on)
- Patient Name
- Date of Birth
- Patient ID
- Contact Number
- Email Address
- Doctor
- Appointment Date
- Appointment Time
- Reason for Visit
- Allergies (list of allergies or "None known")
- Special Notes (blank if none provided)

Data Storage Scope

This app uses Local Storage to persist appointments during a single browser session. You do not need to implement bulk export, historical reporting, or multi-user data sync. Each browser session is independent. If a user clears their browser cache or uses a different browser, previous appointments will not be visible. This is acceptable for this assignment.

You may map one of your custom features to data management, such as enhanced tracking or extra fields that support your Canvas problem.

COMPONENT 3: CALL QUEUE DASHBOARD

Mandatory Requirements

Create a dashboard that shows appointments, focusing on today's work.

The table must show at least:

- Appointment ID
- Patient Name
- Date of Birth (or age, if you choose to compute it)
- Patient ID
- Contact Number
- Doctor
- Appointment Date and time
- Allergies
- Reason for Visit

The dashboard must support:

- Sorting appointments by appointment date and time
- Sorting appointments by doctor
- Filtering appointments by doctor using a dropdown
- Filtering appointments by appointment date using a date picker
- Live updates of the table when filters or sort options change
- Display of the count of appointments for the selected date (for example "Appointments today: 12")

Delete Function

The dashboard must include controls for deleting appointments by Appointment ID.

Delete function:

- Provide a text input where staff enter an Appointment ID
- Provide a Delete button that removes the matching appointment
- If the Appointment ID does not exist, show an error message (for example, "Appointment APT003 not found")
- After successful deletion, update the dashboard immediately

COMPONENT 4: CONFLICT DETECTION

Mandatory Requirements

Prevent double-booking for each doctor.

When staff try to save an appointment:

- Check if the selected doctor already has an appointment at the chosen date and time
- If a conflict exists:
    - Show an error message such as: "Error: Dr. Smith is already booked at 2:00 PM on 2026-03-15. Try 2:30 PM, 3:00 PM, or 3:30 PM."
    - Suggest alternate times based on nearby available slots, or provide a fixed list of suggestions (such as 2:30 PM, 3:00 PM, and 3:30 PM) if calculating availability is beyond your scope
    - Allow staff to select one of the suggested times or enter a different time and retry
- If no conflict exists:
    - Save the appointment normally

You may place one of your custom features in the conflict detection area if it helps address the problem from your Canvas.

COMPONENT 5: ERROR HANDLING AND USER FEEDBACK

Mandatory Requirements

When something goes wrong:

- Every validation failure and blocked operation must trigger an alert() with a specific message
- Messages should match the wording in the Data Validation table or relevant logic
- Avoid generic wording such as "Invalid input"
- On successful save, show "Appointment saved successfully. Ready for next call."
- On successful delete, show "Appointment deleted."
- If an Appointment ID is not found, show "Appointment [ID] not found."

Messages must occur immediately after the user action so staff know what happened.

CUSTOM FEATURES REQUIREMENTS (AT LEAST THREE FEATURES TOTAL)

You must design and implement at least three custom features across Components 1-4. Refer to the Custom Features: Concrete Examples section above for ideas organised by difficulty level.

Rules

- Minimum of three distinct custom features for full marks in this section
- Each feature must connect clearly to the specific intake problem from your Canvas
- You may place features in any combination of:
    - Phone Intake Form
    - Data Management
    - Call Queue Dashboard
    - Conflict Detection
- You may place more than one custom feature in a single component if you wish
- You must describe these features in Part 1, Section 4 (Functional Requirements) and then deliver them in your app
- You may implement more than three features if you wish; additional features show ambition and deep engagement with the problem

SUBMISSION REQUIREMENTS

Zip File Contents

- index.html: Main entry point for the app
- style.css: Styles for the entire app
- script.js: JavaScript for all logic and data management
- Design Thinking Canvas: PowerPoint or PDF with eight to twelve slides

CODE ORGANISATION IN JAVASCRIPT

Follow these standards:

- Use section comments such as:
    - // Constructor functions

    - // Event listeners
    - // Data management functions
    - // Validation helpers
- Use descriptive variable and function names
- Keep functions focused on single tasks
- Add comments to explain any logic that could confuse a new developer

GRADING PHILOSOPHY: HOW YOUR WORK IS EVALUATED

All sections use the same four performance levels.

- **Exceeds Expectations (80-100%):** You met all requirements and added thoughtful improvements based on course concepts. Work at this level shows strong understanding and careful application of ideas from the course.
- **Meets Expectations (60-80%):** You completed what the assignment asked for and followed the instructions. The work is correct and usable, with only minor issues.
- **Does Not Meet Expectations (0-60%):** You missed important parts of the assignment or produced work that does not reach the required standard.
- **Did Not Complete (0 points):** You did not submit this part of the assignment in a usable form.

RUBRIC

PART 1: DESIGN THINKING CANVAS (25%)

Section	Exceeds Expectations (80-100%)	Meets Expectations (60-80%)	Does Not Meet Expectations (0-60%)	Did Not Complete (0%)	Weight
Context	Clear and detailed clinic profile with specific name, location, and size. Realistic workflow description with concrete call volume, peak times, and patient demographics. Stakeholder needs organised separately from pressures with strong insights connected to intake work.	Solid clinic description with a clear workflow, approximate call details, and basic stakeholder needs identified. Minor gaps in detail that do not undermine the overall picture.	Generic or vague clinic description with a weak link to intake challenges. Limited sense of who is affected or why the current system is problematic.	Context section missing or unusable.	6.25%
Problem	One precise WHO-WHAT-CONSTRAINT problem statement. Two or more supporting evidence types. A complete 5-Why chain leading to a clear root cause. Quantified impacts tied directly to phone intake operations.	Clear problem statement with some supporting evidence. A basic 5-Why analysis that reaches a reasonable root cause. At least one meaningful metric quantifying the problem.	Vague or overly broad problem statement. Minimal or missing evidence. Weak, incomplete, or missing 5-Why analysis. No useful metrics to quantify the impact.	No problem statement or analysis provided.	6.25%
Solution	Realistic before-and-after phone call scenario with named staff, specific times, and detailed script highlights. Explicit references to app screens and features used during the call. Measured improvements in speed, accuracy, and completeness that connect back to problem metrics.	Scenario that shows how the app improves calls with some reference to screens and measurable outcomes. Connection between scenario and problem is present but may lack detail.	Generic story with weak or unclear improvements. Missing or superficial connections between the scenario and the app features.	Scenario missing or too unclear to evaluate.	6.25%

Functional Requirements	Custom features clearly named and defined. Each feature directly linked to the Canvas problem with explicit explanation of how it reduces time, errors, or missing data. Estimated measurable impact stated for each feature (time saved, percent accuracy gain, fields auto-filled, or queue improvement).	Custom features that are relevant to the Canvas problem with a clear explanation of purpose and expected outcome. Links to the problem are present but may lack depth or specificity.	Features only loosely linked to the Canvas problem. Descriptions are vague or generic. No measurable impact stated or impact is not credible.	Features missing or completely unrelated to the Canvas problem.	6.25%

PART 2: WEB APPLICATION (75%)

Section	Exceeds Expectations (80-100%)	Meets Expectations (60-80%)	Does Not Meet Expectations (0-60%)	Did Not Complete (0%)	Weight
Phone Intake Form	All required fields and buttons present in the correct order. All validation rules work exactly as specified in the Data Validation table. Error messages match the exact wording from the assignment. Form layout is professional, logically grouped, and easy for staff to follow during live phone calls.	All required elements present with most validation rules working correctly. Layout is clear and usable with minor issues that do not significantly impede workflow.	Missing required fields or buttons. Incorrect or incomplete validation. Error messages missing, incorrect, or confusing. Layout makes the form difficult to use during calls.	Form incomplete or appointments cannot be saved.	7.5%
Data Management	Appointment constructor stores all required data fields. Sequential Appointment IDs (APT001, APT002) generate correctly. Local Storage auto-load on page open and auto-save after each change work reliably. Duplicate bookings blocked	Data stored correctly with all required fields. Local Storage persistence works. Duplicate detection works in most cases. Delete function works. Minor	Incomplete data fields stored. Unreliable Local Storage persistence. Weak or missing duplicate detection. Allergies sometimes empty in saved records.	No working data storage or persistence.	7.5%

	using Patient ID plus doctor plus date plus time rule. Allergies field is never empty in stored records. Delete function removes appointments and updates Local Storage.	issues that do not cause data loss or corruption.	Delete function missing or broken.		
Call Queue Dashboard	Dashboard table displays all required fields including Allergies. Sorting by appointment date and time works. Sorting by doctor works. Filtering by doctor using dropdown works. Filtering by date using date picker works. Table updates live when filters or sort options change. Appointment count displays for the selected date. Delete function accepts Appointment ID input, shows error if ID not found, removes appointment on success, and refreshes dashboard immediately. Layout is clear and clinic-ready.	Dashboard displays appointments with most required fields visible. At least one sorting option and one filtering option work. Delete function works. Layout is readable with minor rough spots.	Missing key columns such as Allergies. Sorting or filtering broken or very limited. Delete function missing, broken, or does not refresh the dashboard. Layout awkward or difficult to scan.	Dashboard missing or not functional.	7.5%
Conflict Detection	Conflict detection catches all double-bookings for the same doctor at the same date and time. Error message clearly identifies the conflict (doctor, date, time). Suggested alternative times provided (calculated). Staff can retry	Conflict detection works for most cases. Error message identifies the conflict. Suggested alternative times provided (fixed list). Retry process works but may have minor friction.	Conflict detection misses some double-bookings. Error messages confusing or missing key details. No alternative times suggested. Retry process awkward or causes data loss.	Conflict detection missing or not functional.	7.5%

	with a different time smoothly without losing entered data.				
Error Handling and User Feedback	Every validation failure triggers a specific alert with wording that matches the Data Validation table. On successful save, displays "Appointment saved successfully. Ready for next call." On successful delete, displays "Appointment deleted." If Appointment ID not found, displays "Appointment [ID] not found." All messages appear immediately after the user action.	Most validation failures show clear, specific alerts. Success and error messages present for key actions. Minor wording variations that do not cause confusion.	Alerts vague, generic, or missing in key places. Success or error feedback missing for important actions. Messages do not help staff understand what happened.	No meaningful feedback provided for any action.	7.5%
Design and Functionality	Page layout follows the natural order of a phone booking call. Information grouped logically (patient details, appointment details, notes). Professional healthcare colour scheme (blues, greens, greys). Clear sans-serif fonts. Appropriate spacing and visual hierarchy. Buttons easy to identify and click. No visual clutter or distracting decoration. Staff can work at speed without confusion.	Layout is organised and readable. Colour scheme and typography are appropriate. Minor rough spots in spacing or grouping that do not significantly impede use.	Layout cluttered or fields out of logical order. Colour scheme or typography unprofessional or hard to read. Buttons hard to find or click. Workflow feels slow or confusing.	Design or behaviour does not support basic booking tasks.	7.5%
Custom Features and Canvas Alignment	The equivalent of at least one Hard, one Medium, and one Easy feature delivered. Each feature clearly linked to the Canvas problem with strong business justification and	Minimum feature requirements met using valid substitutions if needed. Features connect to the Canvas problem with	Feature requirements not met or substitutions invalid. Features shallow, duplicated, or weakly implemented. Weak	No meaningful custom features delivered or no	30%

	measurable impact. Sound technical implementation. Features integrate smoothly without disrupting mandatory functionality. Strong alignment between Canvas and implemented app.	reasonable business justification. Features work with minor bugs. Reasonable alignment between Canvas and implemented app.	business justification. Poor alignment between Canvas and implemented app.	connection to Canvas problem.	