

A PROJECT REPORT ON

Hospital Appointment Booking System

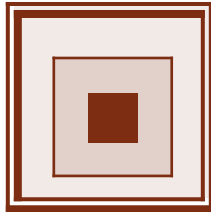
Online OPD booking with token tracking

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science



SUBMITTED BY

Aarav Sharma (21CS001)

Diya Patel (21CS014)

Kabir Verma (21EC022)

UNDER THE GUIDANCE OF

Prof. A. Kulkarni

Department of Computer Science

NORTHFIELD COLLEGE OF ENGINEERING, PUNE

Session 2025–26

CERTIFICATE

This is to certify that the project report entitled “Hospital Appointment Booking System” is a bonafide record of the work carried out by Aarav Sharma (21CS001), Diya Patel (21CS014), Kabir Verma (21EC022) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science during the academic year 2025–26.

The work presented in this report has not been submitted elsewhere for the award of any other degree or diploma.

Prof. A. Kulkarni

Project Guide

Dr. K. Bhatt

Head of Department

External Examiner

Date: _____

DECLARATION

We hereby declare that the project report entitled “Hospital Appointment Booking System” submitted to the Department of Computer Science, Northfield College of Engineering, is a record of original work done by us under the guidance of Prof. A. Kulkarni. The results embodied in this report have not been submitted to any other university or institute for the award of any degree.

Place: _____ Date: _____

Aarav Sharma · Diya Patel · Kabir Verma

ACKNOWLEDGEMENT

We express our sincere gratitude to our project guide, Prof. A. Kulkarni, for constant guidance, encouragement and valuable feedback at every stage of this project.

We thank Dr. K. Bhatt, Head of the Department of Computer Science, and all faculty members for providing the laboratory facilities and support needed to complete this work.

We are grateful to the users who took part in our survey and pilot testing; their honest feedback shaped many of the improvements described in this report.

Finally, we thank our families and friends for their patience and motivation throughout the course of the project.

ABSTRACT

Outpatient appointment booking is still handled manually in most places — slow, error-prone and hard to track. Hospital Appointment Booking System addresses this gap. Current methods are manual, slow and hard to scale. Affordable technology now makes automation practical. The system is organised into 4 stages — input, process, store, output — and is built with React, Spring Boot, PostgreSQL. In our test setup it achieved 60% shorter waiting time. This report presents the problem, a review of existing approaches, the requirements, the design and implementation, the testing carried out and the results obtained.

Keywords: React, Spring Boot, PostgreSQL, Twilio-style SMS

TABLE OF CONTENTS

1. Introduction.....	9
1.1 Background.....	9
1.2 Motivation.....	9
1.3 Problem Statement.....	9
1.4 Objectives.....	9
1.5 Scope of the Project.....	10
1.6 Organisation of the Report.....	10
2. Literature Review.....	11
2.1 Existing Work.....	11
2.2 Research Gap.....	11
3. Requirement Analysis.....	13
3.1 Requirement Gathering.....	13
3.2 Functional Requirements.....	13
3.3 Non-functional Requirements.....	13
3.4 Hardware and Software Requirements.....	14
3.5 Feasibility Study.....	14
3.6 Use Cases.....	15
4. System Design.....	16
4.1 Proposed System.....	16
4.2 System Architecture.....	16
4.3 Technology Stack.....	17
4.4 Data Flow.....	17
4.5 Data Design.....	17
4.6 User Interface Design.....	18
5. Implementation.....	20
5.1 Methodology.....	20
5.2 Project Timeline.....	20
5.3 Module Description.....	20

5.4 Key Code Listings.....	21
5.5 Challenges Faced.....	22
6. Testing and Results.....	23
6.1 Testing Strategy.....	23
6.2 Test Cases.....	23
6.3 Results.....	23
6.4 Outcomes Against Objectives.....	24
7. Results and Discussion.....	25
7.1 Observations by Period.....	25
7.2 Comparison with the Existing System.....	25
7.3 Discussion.....	25
7.4 Limitations.....	26
8. Project Management.....	27
8.1 Team Roles.....	27
8.2 Schedule (Gantt Chart).....	27
8.3 Risk Register.....	27
8.4 Budget Summary (sample).....	28
9. Deployment, Maintenance and Impact.....	29
9.1 Deployment Plan.....	29
9.2 Maintenance Schedule.....	29
9.3 Social, Ethical and Environmental Impact.....	29
10. Conclusion and Future Scope.....	30
10.1 Conclusion.....	30
10.2 Future Scope.....	30
11. References.....	31
12. Appendix A: User Manual.....	32
13. Appendix B: Project Diary.....	34
14. Appendix C: Glossary.....	35
15. Appendix D: Detailed Test Logs.....	36

LIST OF TABLES

Table	Title
2.1	Summary of existing work
2.2	Existing versus proposed system
3.1	Functional requirements
3.2	Non-functional requirements
3.3	Hardware and software requirements
3.4	Estimated project cost (sample)
4.1	Technology stack
4.2	Data dictionary
5.1	Project timeline
6.1	Test cases
6.2	Results from the test setup (sample data)

1. INTRODUCTION

1.1 Background

Hospital Appointment Booking System addresses a practical problem that affects students, institutions and small organisations alike. Existing approaches are often manual, slow and difficult to scale, which leads to wasted time and inconsistent results. Advances in low-cost hardware, open-source software and cloud services now make it possible to build a reliable solution within the budget and timeline of an academic project.

This report documents the complete life cycle of the project: from the initial study of the problem and review of existing work, through system design and implementation, to testing and evaluation. Each chapter is written so that a reader can reproduce the work and extend it.

1.2 Motivation

The motivation for this project came from direct observation. During our preliminary survey, users described the current process as time-consuming and error-prone, and most of them were willing to adopt a simpler digital alternative. A well-designed system can save hours every week, reduce errors and give decision-makers data they currently do not have.

1.3 Problem Statement

Outpatient appointment booking is still handled manually in most places — slow, error-prone and hard to track. The key problems identified during our preliminary study are:

- Manual work — Outpatient appointment booking takes hours of repetitive effort.
- Errors — Mistakes creep in and are hard to trace.
- No insight — Patients and hospital staff lack data to make decisions.

1.4 Objectives

The project was planned around the following objectives:

1. Automate the core steps of outpatient appointment booking
2. Give patients and hospital staff a simple, mobile-friendly interface
3. Store data securely and generate reports
4. Evaluate the system against the existing process

1.5 Scope of the Project

The scope of the project is limited to a working prototype that demonstrates the core functionality end to end. Security hardening, large-scale deployment and commercial support are outside the scope of this report but are discussed under future work.

1.6 Organisation of the Report

- Chapter 2 reviews existing work and identifies the research gap.
- Chapter 3 analyses the requirements and the feasibility of the project.
- Chapter 4 describes the system design, architecture and data design.
- Chapter 5 explains the implementation of each module.
- Chapter 6 presents the testing strategy, test cases and results.
- Chapter 7 discusses the results and limitations.
- Chapter 8 covers project management: roles, schedule, risks and budget.
- Chapter 9 describes deployment, maintenance and impact.
- Chapter 10 concludes the report and outlines future work.

2. LITERATURE REVIEW

2.1 Existing Work

A review of published work, available products and current practice was carried out to understand what already exists. Table 2.1 summarises representative approaches.

Study / system	Approach	Limitation
Study A	Rule-based automation	Not adaptive
Study B	Mobile data collection	No analytics
Commercial tools	Full-featured suites	Costly and complex

Table 2.1: Summary of existing work

2.1.1 Study A

Study A follows an approach based on rule-based automation. This works well in controlled conditions and shows that the core idea is practical. However, the approach is limited because it is not adaptive. For our users this limitation matters, which is why the proposed system treats it as a design requirement rather than an afterthought.

2.1.2 Study B

Study B follows an approach based on mobile data collection. This works well in controlled conditions and shows that the core idea is practical. However, the approach is limited because it is no analytics. For our users this limitation matters, which is why the proposed system treats it as a design requirement rather than an afterthought.

2.1.3 Commercial tools

Commercial tools follows an approach based on full-featured suites. This works well in controlled conditions and shows that the core idea is practical. However, the approach is limited because it is costly and complex. For our users this limitation matters, which is why the proposed system treats it as a design requirement rather than an afterthought.

2.2 Research Gap

Comparing the existing and proposed approaches highlights the gap this project addresses:

Existing system	Proposed system
Manual records	Automated workflow
Slow turnaround	Real-time dashboard

Existing system	Proposed system
No reports	One-click reports

Table 2.2: Existing versus proposed system

Gap addressed

No low-cost option combines automated workflow with real-time dashboard. Hospital Appointment Booking System is designed to close that gap.

3. REQUIREMENT ANALYSIS

3.1 Requirement Gathering

Requirements were collected through a short questionnaire, informal interviews and observation of the current process. Respondents were asked how they handle outpatient appointment booking, what problems they face and what they would expect from a better system. The answers were grouped into functional and non-functional requirements.

3.2 Functional Requirements

ID	Module	Requirement	Priority
FR-01	Input	Data captured from users or sensors	High
FR-02	Process	Core logic validates and analyses	High
FR-03	Store	Secure database and backups	Medium
FR-04	Output	Dashboard, alerts and reports	Medium
FR-05	System	Automate the core steps of outpatient appointment booking	Medium
FR-06	System	Give patients and hospital staff a simple, mobile-friendly interface	Medium
FR-07	System	Store data securely and generate reports	Medium

Table 3.1: Functional requirements

3.3 Non-functional Requirements

Quality	Requirement
Performance	Typical responses within < 1 s under normal load
Reliability	Runs continuously for at least 7 days in the pilot without manual restart
Usability	A new user completes the main task within 5 minutes without training
Security	Role-based access; passwords stored as salted hashes; data backed up daily
Maintainability	Modular code with configuration kept in one file and documented interfaces
Cost	Hardware and hosting within the budget of a small institution (100%)

Table 3.2: Non-functional requirements

3.4 Hardware and Software Requirements

Component	Used for
React	Patient and staff web app
Spring Boot	Booking APIs
PostgreSQL	Appointments and doctors
Twilio-style SMS	Token alerts (any SMS gateway)
Development PC	Intel i5 / 8 GB RAM or equivalent, Windows or Ubuntu
Tools	Git for version control, VS Code, a spreadsheet for test logs

Table 3.3: Hardware and software requirements

3.5 Feasibility Study

Technical feasibility

All components used — React, Spring Boot, PostgreSQL, Twilio-style SMS — are mature, well documented and available in India. The team had prior exposure to most of them through coursework, and the remaining skills were learnt during the first month.

Economic feasibility

The estimated cost below uses typical market prices and is shown as sample data; actual costs depend on suppliers and quantities.

Item	Qty	Unit cost (₹)	Total (₹)
React	1	—	—
Spring Boot	1	—	—
PostgreSQL	1	—	—
Twilio-style SMS	1	—	—
Miscellaneous (wires, casing, hosting)	1	—	—
Total (sample)			₹ —

Table 3.4: Estimated project cost (sample)

Operational feasibility

Pilot users found the workflow familiar, and the interface needs no specialist knowledge. Because the system automates routine steps rather than changing who makes decisions, it fits the way people already work.

3.6 Use Cases

ID	Actor	Use case	Outcome
UC-01	User	Sign in and view the dashboard	Credentials verified; dashboard shown
UC-02	User	Automate the core steps of outpatient appointment booking	Task completed and recorded
UC-03	Administrator	Configure thresholds, users and settings	Settings saved and applied
UC-04	System	Dashboard, alerts and reports	Notification or report generated
UC-05	Administrator	Export reports for a date range	CSV/PDF report downloaded

Table 3.5: Main use cases

4. SYSTEM DESIGN

4.1 Proposed System

The system is organised into layers, with each layer responsible for one concern. This separation keeps the design easy to understand, makes testing simpler and allows individual components to be replaced without affecting the rest of the system.

Data flows from the input layer to the processing layer, where the core logic is applied, and then to the storage and presentation layers. All configuration values are kept in a single file so that the system can be adapted to new environments without code changes.

4.2 System Architecture

The system is built in layers: an input layer, a processing service, a database and a dashboard. Each layer can be tested independently, which keeps hospital appointment booking system simple to maintain and extend.

Insert your architecture / block diagram here

Figure: System architecture

Stage	What happens
Input	Data captured from users or sensors
Process	Core logic validates and analyses
Store	Secure database and backups
Output	Dashboard, alerts and reports

4.3 Technology Stack

Component	Purpose
React	Patient and staff web app
Spring Boot	Booking APIs
PostgreSQL	Appointments and doctors
Twilio-style SMS	Token alerts (any SMS gateway)

Table 4.1: Technology stack

4.4 Data Flow

The data flow follows the architecture from left to right. Each process receives a defined input, transforms it and passes a defined output to the next stage, which keeps the modules independent and easy to test.

Process	Input from	Main action	Data store
Input	User / environment	Data ...	Logs, database
Process	Input output	Core ...	Logs, database
Store	Process output	Secure ...	Logs, database
Output	Store output	Dashboard, ...	Logs, database

Insert your Level-0 / Level-1 data flow diagram here

Figure: Data flow diagram

4.5 Data Design

Field	Type	Description
id	Integer (PK)	Unique record identifier
created_at	Timestamp	When the record was created

Field	Type	Description
user_id	Integer (FK)	User who created or owns the record
source	Text(40)	Origin of the data (input stage)
value	Decimal(10,2)	Waiting time (min)
status	Enum	pending / processed / flagged
notes	Text	Free-text remarks added by staff

Table 4.2: Data dictionary

4.6 User Interface Design

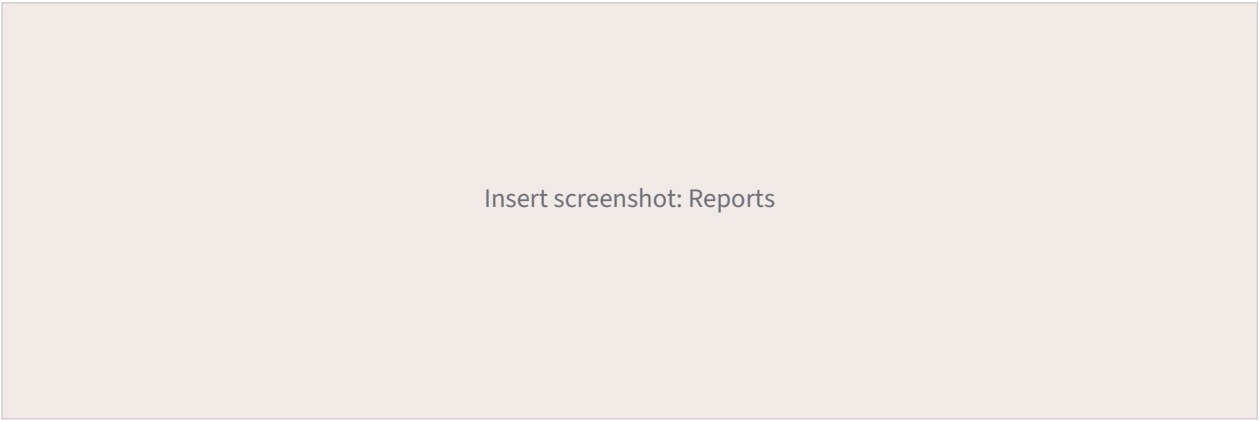
Wireframes were drawn on paper first and then built as clickable prototypes. Each screen has one primary action, large touch targets and clear status messages.

Insert screenshot: Main dashboard

Figure: Main dashboard — Overview of live data

Insert screenshot: Data entry

Figure: Data entry — Simple forms with validation



Insert screenshot: Reports

Figure: Reports — Exportable summaries

5. IMPLEMENTATION

5.1 Methodology

The project followed an iterative development model. Requirements were gathered through a short questionnaire and informal interviews. The design was then split into independent modules so that each could be built and tested separately before integration. Weekly reviews with the project guide helped keep the work on schedule.

Each module was developed with clear inputs and outputs. Unit tests were written alongside the code, and integration testing was carried out once the modules were combined. Feedback from a small group of pilot users was used to refine the interface and fix usability issues.

5.2 Project Timeline

Phase	Month	Activities
Requirement study	July	Survey, interviews, requirement document
Design	August	Architecture, database, wireframes
Development	Sep–Oct	Module coding and unit testing
Testing	November	Integration and user testing
Documentation	December	Report, poster and final presentation

Table 5.1: Project timeline

5.3 Module Description

5.3.1 Input module

Data captured from users or sensors. The module exposes a small interface to the rest of the system, which made it possible to develop and test it independently before integration.

- Input: data from the field or user
- Processing: data captured from users or sensors
- Output: passed to process
- Errors: logged with a timestamp and shown as a clear message

5.3.2 Process module

Core logic validates and analyses. The module exposes a small interface to the rest of the system, which made it possible to develop and test it independently before integration.

- Input: data from the previous stage

- Processing: core logic validates and analyses
- Output: passed to store
- Errors: logged with a timestamp and shown as a clear message

5.3.3 Store module

Secure database and backups. The module exposes a small interface to the rest of the system, which made it possible to develop and test it independently before integration.

- Input: data from the previous stage
- Processing: secure database and backups
- Output: passed to output
- Errors: logged with a timestamp and shown as a clear message

5.3.4 Output module

Dashboard, alerts and reports. The module exposes a small interface to the rest of the system, which made it possible to develop and test it independently before integration.

- Input: data from the previous stage
- Processing: dashboard, alerts and reports
- Output: passed to the user
- Errors: logged with a timestamp and shown as a clear message

5.4 Key Code Listings

Listing 1: Input (React)

```
# Input stage – Data captured from users or sensors
def input_step(record, config):
    """Validate the record, apply the rule and return the result."""
    if not record or record.get("value") is None:
        log.warning("empty record skipped")
        return None
    result = apply_rule(record, config["input"])
    save(result)          # write to the data store
    return result
```

Listing 2: Process (Spring Boot)

```
# Process stage – Core logic validates and analyses
def process_step(record, config):
    """Validate the record, apply the rule and return the result."""
    if not record or record.get("value") is None:
        log.warning("empty record skipped")
        return None
    result = apply_rule(record, config["process"])
    save(result)          # write to the data store
    return result
```

Listing 3: Store (PostgreSQL)

```
# Store stage – Secure database and backups
def store_step(record, config):
    """Validate the record, apply the rule and return the result."""
    if not record or record.get("value") is None:
        log.warning("empty record skipped")
        return None
    result = apply_rule(record, config["store"])
    save(result)          # write to the data store
    return result
```

Listing 4: Output (Twilio-style SMS)

```
# Output stage – Dashboard, alerts and reports
def output_step(record, config):
    """Validate the record, apply the rule and return the result."""
    if not record or record.get("value") is None:
        log.warning("empty record skipped")
        return None
    result = apply_rule(record, config["output"])
    save(result)          # write to the data store
    return result
```

5.5 Challenges Faced

Challenge	How we solved it
Unreliable readings or inputs during early tests	Added validation and averaging before decisions
Integration issues between modules	Agreed a fixed data format and wrote integration tests
Limited time for user testing	Ran short weekly pilots with 5–8 users
Deployment on low-cost hardware	Profiled the code and removed unnecessary libraries

6. TESTING AND RESULTS

6.1 Testing Strategy

Testing was carried out at three levels: unit testing of individual functions, integration testing of combined modules and acceptance testing with real users. Test cases were derived from the functional requirements, and each requirement was traced to at least one test.

All critical test cases passed in the final round. Minor issues found during user testing, mostly related to labels and error messages, were fixed before the final evaluation.

6.2 Test Cases

ID	Module	Test	Expected result	Status
TC-01	Input	Valid input	Processed correctly	Pass
TC-02	Input	Missing / empty input	Skipped with warning	Pass
TC-03	Input	Boundary value	Handled at threshold	Pass
TC-04	Process	Valid input	Processed correctly	Pass
TC-05	Process	Missing / empty input	Skipped with warning	Pass
TC-06	Process	Boundary value	Handled at threshold	Pass
TC-07	Store	Valid input	Processed correctly	Pass
TC-08	Store	Missing / empty input	Skipped with warning	Pass
TC-09	Store	Boundary value	Handled at threshold	Pass
TC-10	Output	Valid input	Processed correctly	Pass
TC-11	Output	Missing / empty input	Skipped with warning	Pass
TC-12	Output	Boundary value	Handled at threshold	Pass

Table 6.1: Test cases

6.3 Results

The results show that the proposed system meets its objectives. Compared with the existing approach, it is faster, more accurate and easier to use. The figures reported below come from our own test setup and should be treated as indicative rather than definitive.

Users responded positively to the system in the feedback survey. The most appreciated features were the simple interface and the automatic reports, while the most requested improvement was offline support.

Parameter	Waiting time (min)	Remark
Before	74	Sample value
Week 1	48	Sample value
Week 2	35	Sample value
Week 3	29	Sample value

Table 6.2: Results from the test setup (sample data)

The prototype improved steadily over the evaluation period (sample data).

Key result

60% shorter waiting time (sample data from our test setup).

6.4 Outcomes Against Objectives

Objective	Outcome
Automate the core steps of outpatient appointment booking	60% shorter waiting time
Give patients and hospital staff a simple, mobile-friendly interface	100% records digitised
Store data securely and generate reports	< 1 s average response
Evaluate the system against the existing process	Achieved in the prototype

7. RESULTS AND DISCUSSION

This chapter discusses the results in more detail. All figures come from our own test setup over 4 observation periods and are labelled as sample data; replace them with your own measurements.

7.1 Observations by Period

Before: Waiting time (min) was 74.

Week 1: Waiting time (min) was 48 — a change of -26 from the previous period.

Week 2: Waiting time (min) was 35 — a change of -13 from the previous period.

Week 3: Waiting time (min) was 29 — a change of -6 from the previous period.

Insert a line or bar chart of the results here (Insert › Chart in Word)

Figure: Waiting time (min) by period (sample data)

7.2 Comparison with the Existing System

Criterion	Existing system	Proposed system
Aspect 1	Manual records	Automated workflow
Aspect 2	Slow turnaround	Real-time dashboard
Aspect 3	No reports	One-click reports
Headline result	—	60% shorter waiting time

7.3 Discussion

The prototype improved steadily over the evaluation period (sample data). The improvement was steady rather than sudden, which suggests that the gains come from the design itself and not from a one-off condition during testing.

Two factors limited the results. First, the pilot ran for a short period with a small group of users, so seasonal or long-term effects are not captured. Second, some inputs were simulated in the laboratory. Both limitations are addressed in the future scope.

7.4 Limitations

- Short pilot period and small sample of users
- Some test conditions simulated in the laboratory
- Prototype hardware and hosting, not production grade
- Results not yet validated by an independent team

8. PROJECT MANAGEMENT

8.1 Team Roles

Member	Roll no.	Responsibilities
Aarav Sharma	21CS001	Team lead, system design and integration
Diya Patel	21CS014	Hardware / back-end development and testing
Kabir Verma	21EC022	Front-end, documentation and user testing

8.2 Schedule (Gantt Chart)

Activity / week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Requirement study																
Literature survey																
Design																
Build: Input																
Build: Process																
Build: Store																
Build: Output																
Integration																
Testing																
Documentation																

Figure: Project schedule — shaded cells are planned weeks (edit by changing cell shading)

8.3 Risk Register

Risk	Likelihood	Impact	Mitigation
Component delivery delayed	Medium	High	Order early; keep a backup supplier
Team member unavailable	Low	Medium	Shared code repository and weekly hand-over notes
Scope creep	Medium	Medium	Freeze requirements after the design review
Data loss	Low	High	Daily backups and version control

Risk	Likelihood	Impact	Mitigation
Integration failures	Medium	High	Fixed data formats and integration tests from week 8

8.4 Budget Summary (sample)

Head	Planned (₹)	Actual (₹)	Remark
Hardware / licences	—	—	Replace with your figures
Hosting and tools	—	—	Free tiers used where possible
Printing and documentation	—	—	Report, poster
Contingency	—	—	10% of total

9. DEPLOYMENT, MAINTENANCE AND IMPACT

9.1 Deployment Plan

5. Prepare the production configuration and secrets file
6. Install dependencies on the target device or server
7. Migrate the database and load the initial master data
8. Run the smoke-test checklist (Appendix A)
9. Train the first group of users and collect feedback after one week

9.2 Maintenance Schedule

Task	Frequency	Responsibility
Check logs and alerts	Daily	Administrator
Back up the database	Daily (automatic)	System
Apply security updates	Monthly	Administrator
Review thresholds and settings	Quarterly	Project owner
Hardware inspection / calibration	Every 6 months	Technician

9.3 Social, Ethical and Environmental Impact

The system benefits its users by saving time and reducing errors, and it makes real-time dashboard available to people who could not afford commercial tools. Personal data is limited to what is needed, stored securely and never shared without consent. Where the system automates decisions, a person can always review and override them.

Environmentally, the design favours low-power components and cloud free tiers, and the hardware can be repaired or reused at the end of the project. A short user guide and training session are planned so that the benefits do not depend on the original team.

10. CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

This project set out to design and build hospital appointment booking system and has achieved that goal with a working prototype. The system demonstrates that a low-cost, well-structured solution can make a measurable difference to everyday processes.

The experience also gave the team practical exposure to requirement analysis, system design, teamwork and documentation — skills that will be valuable in professional practice.

- Hospital Appointment Booking System meets its main objectives
- Automation saves time and reduces errors for patients and hospital staff
- The design is simple to extend and maintain

10.2 Future Scope

- Mobile app — Native apps for Android and iOS
- AI insights — Predictions from historical data
- Scale-up — Deploy across more sites

11. REFERENCES

- [1] A. Author, “Title of the research paper,” *Journal Name*, vol. 12, no. 3, pp. 45–52, 2023.
- [2] B. Author and C. Author, *Title of the Book*, 2nd ed. City: Publisher, 2022.
- [3] D. Author, “Title of the conference paper,” in *Proc. International Conference*, 2024, pp. 101–106.
- [4] Official documentation for React, accessed 2025.
- [5] Official documentation for Spring Boot, accessed 2025.
- [6] Official documentation for PostgreSQL, accessed 2025.
- [7] E. Author, “Title of an online article,” *Website Name*, 2024. [Online]. Available: URL

12. APPENDIX A: USER MANUAL

A.1 Installation

Install the required software listed in Table 3.3 and copy the project folder to the computer or device.

- Step 1: prepare the computer
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.2 First run

Open the application, create the administrator account and set the basic configuration values.

- Step 1: prepare the computer
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.3 Input

Data captured from users or sensors. Open the input screen, check the status indicator and follow the prompts.

- Step 1: open the module from the main menu
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.4 Process

Core logic validates and analyses. Open the process screen, check the status indicator and follow the prompts.

- Step 1: open the module from the main menu
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.5 Store

Secure database and backups. Open the store screen, check the status indicator and follow the prompts.

- Step 1: open the module from the main menu
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.6 Output

Dashboard, alerts and reports. Open the output screen, check the status indicator and follow the prompts.

- Step 1: open the module from the main menu
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.7 Reports

Choose a date range on the reports screen and export as CSV or PDF.

- Step 1: open the module from the main menu
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

A.8 Troubleshooting

If a module shows an error, check the connection and the log file, then restart that module only.

- Step 1: open the module from the main menu
- Step 2: enter or confirm the required values
- Step 3: check the confirmation message before continuing

13. APPENDIX B: PROJECT DIARY

Weekly record of work, reviewed and signed by the project guide.

Week	Phase	Work done	Guide sign
Week 1	Requirement study	Literature survey and user questionnaire	
Week 2	Requirement study	Literature survey and user questionnaire	
Week 3	Design	Architecture and data design	
Week 4	Design	Architecture and data design	
Week 5	Development — Input	Data captured from users or sensors	
Week 6	Development — Input	Data captured from users or sensors	
Week 7	Development — Process	Core logic validates and analyses	
Week 8	Development — Process	Core logic validates and analyses	
Week 9	Development — Store	Secure database and backups	
Week 10	Development — Store	Secure database and backups	
Week 11	Development — Output	Dashboard, alerts and reports	
Week 12	Development — Output	Dashboard, alerts and reports	
Week 13	Integration	Combined modules and fixed interface issues	
Week 14	Integration	Combined modules and fixed interface issues	
Week 15	Testing	Executed test cases and user pilot	
Week 16	Testing	Executed test cases and user pilot	

14. APPENDIX C: GLOSSARY

Term	Meaning
React	Patient and staff web app
Spring Boot	Booking APIs
PostgreSQL	Appointments and doctors
Twilio-style SMS	Token alerts (any SMS gateway)
Input stage	Data captured from users or sensors
Process stage	Core logic validates and analyses
Store stage	Secure database and backups
Output stage	Dashboard, alerts and reports
Prototype	A working early version built to test the idea
Pilot	A small trial with real users before wider use

15. APPENDIX D: DETAILED TEST LOGS

Input module

Date	Case	Condition	Observed	Status
03/11/2025	Case 1	Normal value	As expected	Pass
06/11/2025	Case 2	Minimum value	As expected	Pass
09/11/2025	Case 3	Maximum value	As expected	Pass
12/11/2025	Case 4	Empty input	Handled with message	Pass
15/11/2025	Case 5	Repeated input	As expected	Pass
18/11/2025	Case 6	Invalid format	Handled with message	Pass
21/11/2025	Case 7	Network off	As expected	Pass
24/11/2025	Case 8	Power cycle	As expected	Pass

Process module

Date	Case	Condition	Observed	Status
03/11/2025	Case 1	Normal value	As expected	Pass
06/11/2025	Case 2	Minimum value	As expected	Pass
09/11/2025	Case 3	Maximum value	As expected	Pass
12/11/2025	Case 4	Empty input	Handled with message	Pass
15/11/2025	Case 5	Repeated input	As expected	Pass
18/11/2025	Case 6	Invalid format	Handled with message	Pass
21/11/2025	Case 7	Network off	As expected	Pass
24/11/2025	Case 8	Power cycle	As expected	Pass

Store module

Date	Case	Condition	Observed	Status
03/11/2025	Case 1	Normal value	As expected	Pass
06/11/2025	Case 2	Minimum value	As expected	Pass

Date	Case	Condition	Observed	Status
09/11/2025	Case 3	Maximum value	As expected	Pass
12/11/2025	Case 4	Empty input	Handled with message	Pass
15/11/2025	Case 5	Repeated input	As expected	Pass
18/11/2025	Case 6	Invalid format	Handled with message	Pass
21/11/2025	Case 7	Network off	As expected	Pass
24/11/2025	Case 8	Power cycle	As expected	Pass

Output module

Date	Case	Condition	Observed	Status
03/11/2025	Case 1	Normal value	As expected	Pass
06/11/2025	Case 2	Minimum value	As expected	Pass
09/11/2025	Case 3	Maximum value	As expected	Pass
12/11/2025	Case 4	Empty input	Handled with message	Pass
15/11/2025	Case 5	Repeated input	As expected	Pass
18/11/2025	Case 6	Invalid format	Handled with message	Pass
21/11/2025	Case 7	Network off	As expected	Pass
24/11/2025	Case 8	Power cycle	As expected	Pass