Software Requirements Specification, System Design Specification and Test Plan

for

Laundry System

Version 1.0

Prepared by

GROUP: Group 4

Ti-Jean Dehaney	**ID #620172199**	**tijean.dehaney@mymona.uwi.edu**
Sean Thompson	**ID #620170129**	**sean.thompson02@mymona.uwi.edu**
Darian Matthews	**ID #621069906**	**Darian.Matthews@mymona.uwi.edu**
Caudillo Jones	**ID #620172731**	**caudillo.jones@mymona.uwi.edu**

Ruth-Ann Allen	ID # 620161125	ruthann.allen@mymona.uwi.edu
Tristan Thompson	ID #620153718	tristanthomp876@gmail.com

Course Instructor:	Mr. Ricardo Anderson
Course:	COMP2140 – Introduction to Software Engineering
Studio Facilitator:	*Claudine Allen*
Date:	October 23, 2025

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Ti-Jean Dehaney, Sean Thompson, Darian Matthews, Caudillo Jones, Ruth-Ann Allen, Tristan Thompson	Initial completed version of the Software Requirements Specification (SRS), System Design Specification (SDS), and test plan for the Taylor Hall Laundry Appointment System.	12/3/25

1 Overall Description

1.1 Product Context and Need

Taylor Hall's laundry system lacks an organized booking process, causing overcrowding, long waits, conflicts over machine use. A digital appointment and ID verification system is needed to streamline scheduling and improve the student experience.

1.2 Product Functionality

The system allows users to add, edit and delete appointments while preventing double bookings. It assigns each user an ID for managing appointments, viewing available time slots and confirming bookings using their appointment ID.

1.3 Stakeholders and Users Characteristics

Primary stakeholders include Taylor Hall residents who book laundry slots and hall administrators who manage schedules and monitor usage. Secondary stakeholders include maintenance staff and the development team responsible for system implementation and upkeep.

1.4 Operating Environment

The proposed laundromat booking system will operate through a web-based platform which will be accessible by both desktop and mobile devices to allow portability. The backend will be linked to a database such as MySQL or PostgreSql and the frontend will be available through web browsers in which an internet connection will be required.

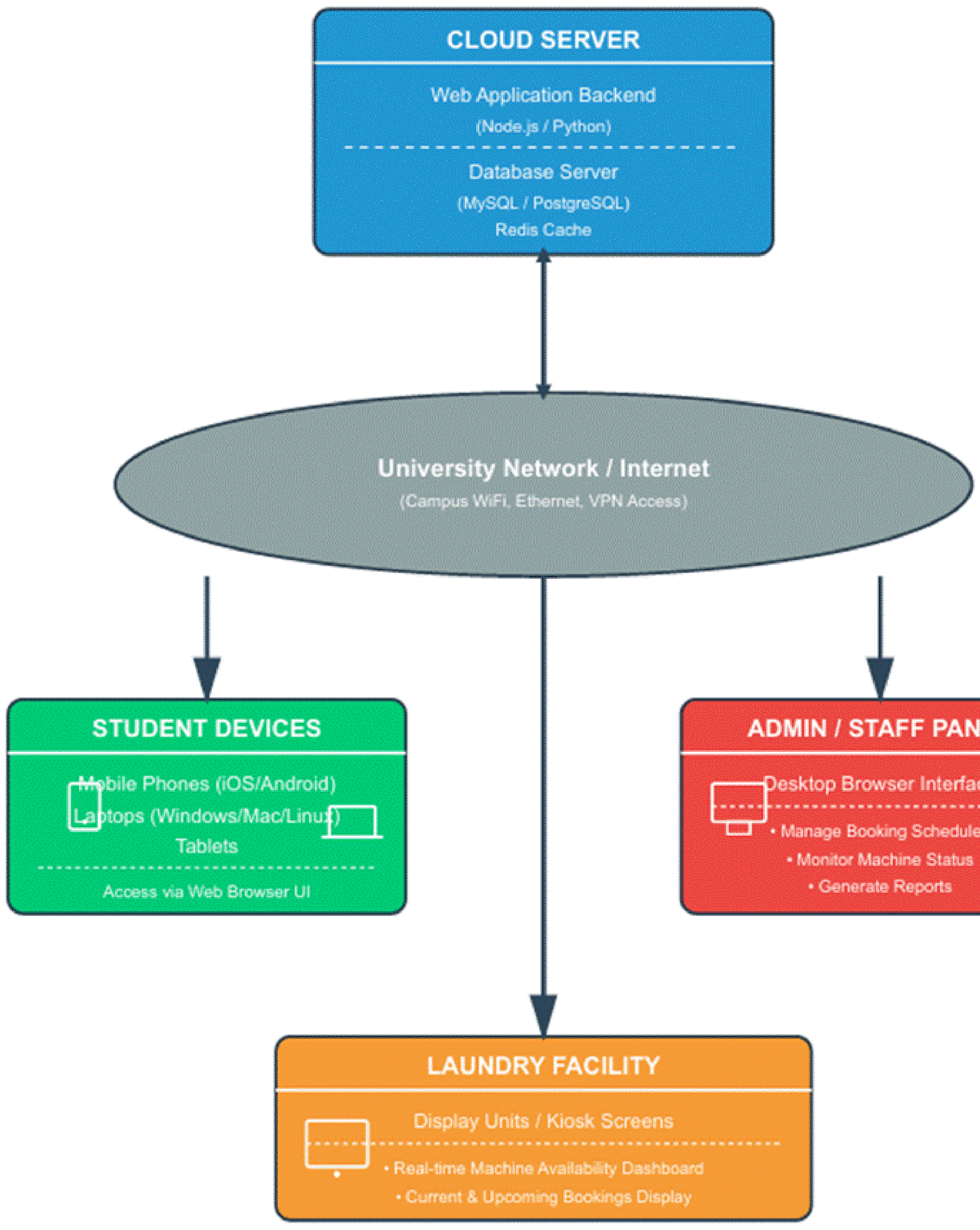
-Desktop/Laptop: Windows 10 or later, macOS 10.14 or later, linux Ubuntu 20.04 or later.

- Mobile Devices: IOS 13+ or Android 8.0 and above

- Memory: 4GB ram minimum to allow smooth browser operation

- Stable internet connection with minimum 2Mbps bandwith

Laundromat Booking System Architecture


CLOUD SERVER
Web Application Backend
(Node.js / Python)
Database Server
(MySQL / PostgreSQL)
Redis Cache
University Network / Internet
(Campus WiFi, Ethernet, VPN Access)
STUDENT DEVICES
Mobile Phones (iOS/Android)
Laptops (Windows/Mac/Linux)
Tablets
Access via Web Browser UI
ADMIN / STAFF PANEL
Desktop Browser Interface
• Manage Booking Schedules
• Monitor Machine Status
• Generate Reports
LAUNDRY FACILITY
Display Units / Kiosk Screens
• Real-time Machine Availability Dashboard
• Current & Upcoming Bookings Display

1.5 Design and Implementation Constraints

- *Limited server resources since we will be constrained to free cloud hosting with maybe a maximum capacity of 100 users which will require optimized code with real-time booking responses.*
- *Preventing double-bookings*
- *Browser and device compatibility as the system must remain functional across the standard browsers and various display sizes for both desktop and mobile*

1.6 Assumptions and Dependencies

- *The laundromat has a fixed number of washing and drying machines.*
- *The server hosting the system will remain and have sufficient uptime and resource*
- *Machine availability data will be updated in real time without delays*
- *The project will function through standard web technologies and proprietary software*
- *Both user and administrators will have stable internet connections to conduct tasks.*
- *All users will access the system through modern and current web browsers and operating systems.*

2 Specific Requirements

2.1 External Interface Requirements

2.1.1 Hardware Interfaces

Resident Devices

- ***Devices:*** *smartphones, tablets, laptops.*
- ***Use:*** *book, reschedule, cancel*
- ***Interaction:*** *web browser, touch or keyboard*

Laundry Room or Tablet

- ***Device:*** *Android or iPad tablet mounted in the laundry room.*
- ***Use:*** *A laundry room worker would ask you for your id number to verify that it is indeed your time for washing*

2.1.2 Software Interfaces

Supported OS:

- ***Windows*** *10 or later.*
- ***macOS*** *12 or later.*
- ***Linux*** *Ubuntu 20.04 or later.*
- ***Android*** *11 or later.*
- ***iOS/iPadOS*** *15 or later.*

Browsers: *latest two versions of Chrome, Edge, Firefox, Safari.*

OS Services the app uses:

- ***Networking:*** *OS TCP/IP stack through the browser.*
- ***Camera:*** *OS camera permission through the browser.*

- ***Notifications:*** *optional web push routed to the OS notification center.*

- ***Storage:*** *local cache with browser* ***IndexedDB*** *and* ***localStorage*** *for session data, no system files.*

- ***Time and locale:*** *read via browser to format dates and times.*

Constraints: *no kernel modules, no admin rights, no native installers.*

2.1.3 Communications Interfaces

- *The Laundry Appointment and ID Verification System will use standard web communication protocols to connect users, the web server, and the database. All interactions between the user's device and the system will occur through a web browser using* ***HTTPS****, ensuring that all data is securely transmitted.*

- *Email and SMS services will be used for appointment confirmations and reminders. Emails will be sent using secure* ***SMTP*** *connections, and SMS messages will be sent through a verified third-party gateway over secure HTTPS connections. All communication will be encrypted to protect user credentials, booking details, and system messages.*

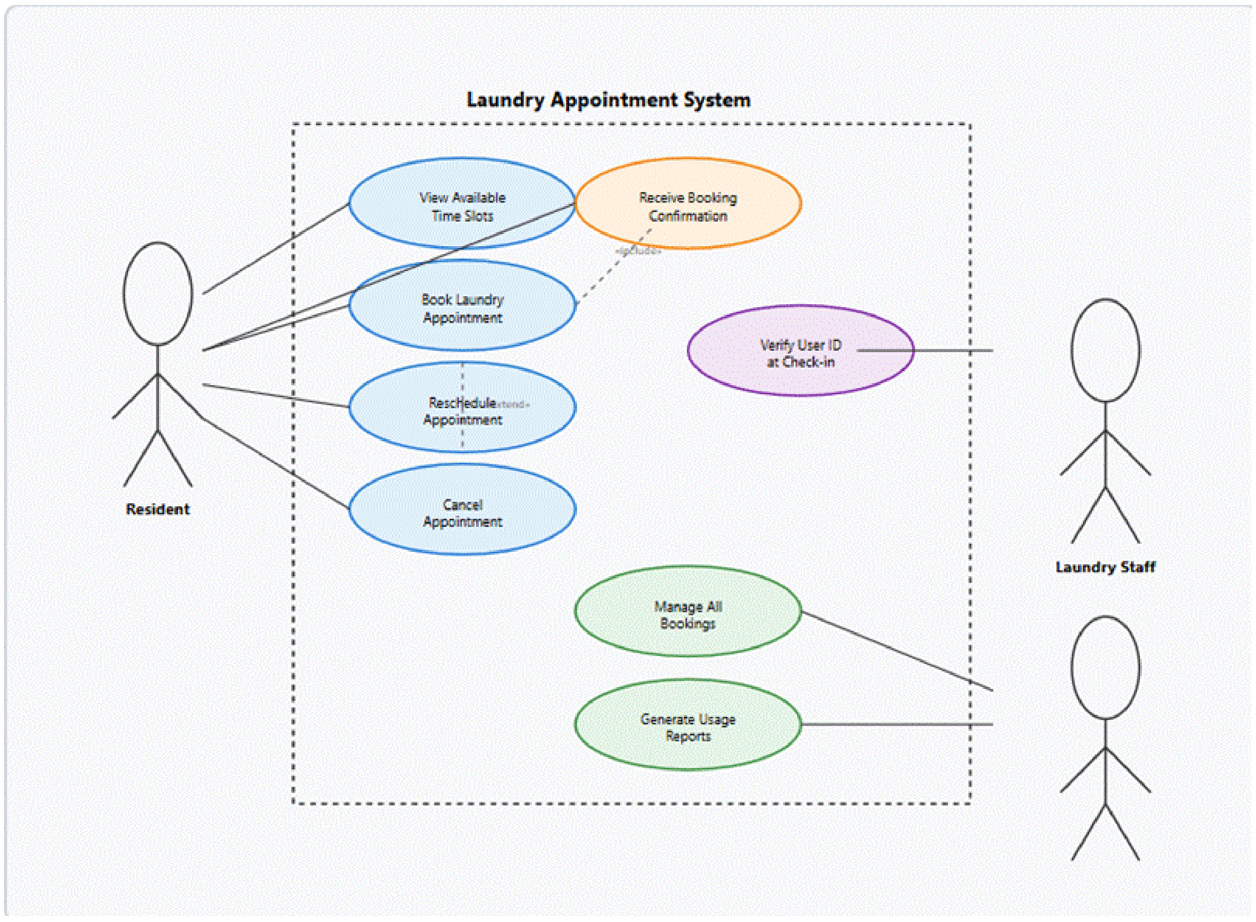
2.2 System Features

Feature ID	Feature Name	User Story	Description / Expected Outcome	Priority	Team Owner
F1	View Available Time Slots	As a resident, I want to view all available washing and drying machine time slots, so that I can choose a convenient time to do my laundry.	The system will display all available and booked slots in a clear schedule format, updating in real time whenever changes occur.	High	Ti-Jean Dehaney
F2	Book Laundry Appointment	As a resident, I want to book a laundry machine at a specific time, so that I can secure a machine when I need it and avoid conflicts with others.	The user can select an available slot and confirm a booking. The system stores the appointment and sends a confirmation message with a unique booking ID.	High	Sean Thompson

F3	Reschedule or Cancel Appointment	As a resident, I want to reschedule or cancel my laundry booking, so that I can make changes if my plans change.	The system allows users to view their bookings and modify or cancel them up to 30 minutes before the start time. The schedule updates automatically.	Medium	Darian Matthews
F4	ID Verification in Laundry Room	As a laundry room staff member, I want to verify users' bookings using their student ID or booking code, so that only authorized residents can use the machines at their booked times.	Staff can check booking validity through a tablet interface. The system confirms or rejects access instantly.	High	Caudillo Jones
F5	Notification and Reminder System	As a resident, I want to receive booking confirmations and reminders, so that I don't forget my scheduled laundry times.	The system sends email or SMS confirmations after booking, and reminders 30 minutes before the scheduled time.	Medium	Ruth-Ann Allen

F6	Administrator Dashboard	As an administrator, I want to view and manage all laundry bookings and reports, so that I can monitor usage, resolve conflicts, and maintain system efficiency.	Admins can view all bookings, edit or delete them if needed, and generate usage reports or export data in CSV/PDF format.	High	Tristan Thompson

2.3 Use Case View



Use Case Narratives

UC1: View Available Time Slots

Actor: Resident

Description: A resident views the schedule of all available and booked laundry machine time slots to determine when they can book a machine.

Precondition: The resident must be logged into the system with valid credentials.

Basic Flow:

1. *Resident logs into the laundry booking system.*
2. *System displays the main dashboard.*
3. *Resident selects "View Schedule" or navigates to the booking page.*
4. *System retrieves current machine availability from the database.*
5. *System displays a calendar or time-slot grid showing available and booked slots.*
6. *Resident views available time slots for washing and drying machines.*

Postcondition: The resident has viewed the current availability and can proceed to book an available slot if desired.

UC2: Book Laundry Appointment

Actor: Resident

Description: A resident books a specific time slot for a washing or drying machine to secure it for their use.

Precondition: The resident is logged in and has viewed available time slots. The selected time slot must be available (not already booked).

Basic Flow:

1. *Resident selects an available time slot from the schedule.*
2. *System displays booking confirmation screen with selected date, time, and machine type.*
3. *Resident confirms the booking.*
4. *System validates that the slot is still available and checks for booking conflicts.*

5. *System creates the appointment record in the database with a unique booking ID.*
6. *System generates and displays a booking confirmation with the booking ID.*
7. *System sends a confirmation notification (email/SMS) to the resident. (includes UC5)*

Postcondition: The appointment is recorded in the system, the time slot is marked as booked, and the resident receives a confirmation with their unique booking ID.

UC3: Reschedule Appointment

Actor: Resident

Description: A resident changes the date or time of an existing laundry appointment to a different available slot.

Precondition: The resident has an existing booking and is logged into the system. The current time must be at least 30 minutes before the scheduled appointment.

Basic Flow:

1. *Resident navigates to "My Bookings" section.*
2. *System displays all current and upcoming appointments for the resident.*
3. *Resident selects the appointment they wish to reschedule.*
4. *System verifies the booking is eligible for rescheduling (at least 30 minutes before start time).*
5. *System displays available alternative time slots.*
6. *Resident selects a new time slot and confirms the change.*
7. *System updates the appointment record with the new date/time.*
8. *System releases the original time slot and marks the new slot as booked.*
9. *System sends an updated confirmation notification to the resident.*

Postcondition: The appointment is updated with the new time, the original slot becomes available again, and the resident receives updated confirmation.

UC4: Cancel Appointment

Actor: Resident

Description: A resident cancels an existing laundry appointment, freeing the time slot for other residents.

Precondition: The resident has an existing booking and is logged into the system. The current time must be at least 30 minutes before the scheduled appointment.

Basic Flow:

1. *Resident navigates to "My Bookings" section.*
2. *System displays all current and upcoming appointments.*
3. *Resident selects the appointment they wish to cancel.*
4. *System verifies the booking is eligible for cancellation (at least 30 minutes before start time).*
5. *System prompts resident to confirm cancellation.*
6. *Resident confirms the cancellation.*
7. *System deletes or marks the appointment as cancelled in the database.*
8. *System releases the time slot, making it available to other residents.*
9. *System sends a cancellation confirmation notification to the resident.*

Postcondition: The appointment is cancelled, the time slot is available for booking, and the resident receives cancellation confirmation.

UC5: Receive Booking Confirmation

Actor: Resident

Description: The system automatically sends booking confirmations and reminders to residents after they book, reschedule, or when their appointment is approaching.

Precondition: A resident has successfully booked or rescheduled an appointment, or has an appointment scheduled within 30 minutes.

Basic Flow:

1. *System triggers notification event (after booking/rescheduling or 30 minutes before appointment).*
2. *System retrieves resident's notification preferences (email/SMS).*
3. *System generates notification message including booking details and unique booking ID.*
4. *System sends notification through selected channel(s).*
5. *Resident receives the confirmation or reminder message.*
6. *System logs the notification delivery status.*

Postcondition: The resident receives confirmation or reminder notification and is informed of their booking details.

UC6: Verify User ID at Check-in

Actor: Laundry Staff

Description: Laundry room staff verifies a resident's booking using their student ID to ensure only authorized users access machines during their reserved time.

Precondition: Staff has access to the verification tablet interface. The resident has arrived at the laundry room with their booking ID or student ID.

Basic Flow:

1. *Resident arrives at laundry room during their scheduled time slot.*
2. *Staff requests the resident's student ID number or booking ID.*
3. *Staff enters the ID into the verification tablet.*
4. *System queries the database for active bookings matching the ID and current time.*
5. *System validates that the booking exists and the current time falls within the scheduled slot.*
6. *System displays verification result (approved or denied) on the tablet.*
7. *Staff grants or denies access to the machine based on the system result.*
8. *System logs the check-in event in the database.*

Postcondition: The resident is either granted access to use the machine or informed their booking is invalid. The verification event is logged.

UC7: Manage All Bookings

Actor: Administrator

Description: An administrator views, edits, or deletes any booking in the system to resolve conflicts, handle complaints, or perform system maintenance.

Precondition: The administrator is logged into the system with administrative privileges.

Basic Flow:

1. *Administrator logs into the admin dashboard.*
2. *Administrator navigates to "Manage Bookings" section.*
3. *System displays a comprehensive list of all bookings (past, current, and upcoming).*
4. *Administrator can filter bookings by date, resident, machine, or status.*
5. *Administrator selects a booking to view details.*
6. *Administrator chooses to edit or delete the booking.*
7. *System prompts for confirmation if deleting.*
8. *Administrator confirms the action.*
9. *System updates or removes the booking record from the database.*

Postcondition: The booking is modified or deleted, affected residents are notified, and the administrative action is logged.

UC8: Generate Usage Reports

Actor: Administrator

Description: An administrator generates reports on laundry system usage to analyze patterns, identify peak times, and make informed decisions about resource allocation.

Precondition: The administrator is logged into the system with administrative privileges. Historical booking data exists in the database.

Basic Flow:

1. *Administrator navigates to "Reports" section in the admin dashboard.*
2. *System displays available report types (usage by day, by resident, by machine, peak times, cancellation rates, etc.).*
3. *Administrator selects report type and specifies parameters (date range, machines, residents).*
4. *System queries the database and aggregates relevant booking data.*
5. *System generates the report with visualizations (charts, tables, statistics).*
6. *System displays the report on screen.*
7. *System generates the export file and provides download link.*
8. *Administrator downloads the report for external use or record-keeping.*

Postcondition: The usage report is generated, viewed by the administrator, and optionally exported for further analysis or documentation

3 Other Non-functional Requirements

3.1 Performance Requirements

Availability and Reliability:

- The system should maintain atleast 99**% uptime** per month, with automatic recovery in case of minor failures.
- *Reasoning:* Laundry scheduling is a daily activity, and downtime can cause missed appointments or overcrowding.

System Response Time:

- All user transactions (such as booking, editing, or canceling an appointment) must complete within short time under normal network conditions.

- *Reasoning:* Users expect fast response when booking or checking time slots, especially during peak hours.

Database Update Efficiency:

- Any updates to appointment records or machine availability should reflect in the system database within **a short time** of user action.

- *Reasoning:* Ensures that real-time scheduling information remains accurate and prevents booking conflicts.

3.2 Safety and Security Requirements

Data Backup and Recovery:

- *The system must perform automatic **daily backups** of all appointment and user data. In the event of a system crash, recovery should be possible within **30 minutes**.*
- *Rationale: Prevents data loss that could disrupt operations or cause double bookings.*

Access Restriction to Authorized Users:

- *Only registered residents and authorized staff can log into the system using valid credentials (e.g., student ID or institutional email).*
- *Reasoning: Ensures unauthorized individuals cannot access the machines when its not their appointment time book or modify appointments.*

3.3 Software Quality Attributes

Maintainability and Design for Change

- ***Goal:*** *ship small changes in under 3 days, hotfix in 4 hours.*

- ***How you will achieve it:*** *services for Booking, Machines, Notifications. Config-driven rules for slot length, per-resident caps, blackout windows. Feature flags for risky changes. API Coding standards, linting, and code reviews. Automated unit and integration tests. Simple logs and error codes.*

2) Usability

- ***Goal:*** *a resident completes a booking in ≤ 60 seconds. Task success ≥ 95% for book, reschedule, cancel, check-in.*

- ***How to achieve it:*** *one-page booking flow. Mobile-first layout. Plain language labels. Inline validation. Clear undo for cancel. Keyboard access.*

3) Security and Privacy

- ***Goal:*** *protect resident data, enforce least privilege.*

- ***How you will achieve it:*** *HTTPS for all traffic. Role-based access for Resident, Staff, Admin. Session timeout after 10 minutes idle. Audit logs for login and booking edits. Input validation, prepared statement*

Appendix

To acquire these requirements a questionnaire was given out to the residents of Taylor Hall on October 11,2025. This questionnaire consisted of 15 questions both open ended and closed ended. This method was chosen as it is cost effective and is a easy way of eliciting requirements.

Questionnaire:

Section A: General Information

1. ***How often do you use the laundry facilities at Taylor Hall?***
   ☐ *Daily*
   ☐ *2–3 times per week*
   ☐ *Once per week*
   ☐ *Occasionally*

2. ***How do you currently schedule or manage your laundry time?***
   ☐ *Walk-in (first come, first served)*
   ☐ *Written sign-up sheet*
   ☐ *Through staff assistance*
   ☐ *Other (please specify):* ________________________

3. ***On a scale of 1–5, how satisfied are you with the current laundry booking process?***
   (1 = Very Dissatisfied, 5 = Very Satisfied)
   ☐ *1* ☐ *2* ☐ *3* ☐ *4* ☐ *5*

4. ***What issues have you experienced with the current booking process?***
   (Open-ended)

4. ____

____

Section B: System Use and Needs

5. ***Do you find it difficult to know when machines are available?***
   ☐ *Yes*
   ☐ *No*
   ☐ *Sometimes*

6. ***If an online booking system were introduced, how likely would you be to use it?***
   ☐ *Very likely*
   ☐ *Somewhat likely*
   ☐ *Neutral*
   ☐ *Unlikely*

7. ***What features would you find most useful in an online laundry appointment system?***
   (Open-ended)

8. ______________________________

9. ***Would you prefer to receive booking confirmations and reminders via:***
   ☐ *Email*
   ☐ *Text message (SMS)*
   ☐ *In-app notification*
   ☐ *Any of the above*

Section C: Performance Expectations

9. ***How long are you willing to wait for a booking confirmation to load?***
   ☐ *Less than 5 seconds*
   ☐ *5–10 seconds*
   ☐ *More than 10 seconds*

10. ***Would you expect the system to be available 24/7, including weekends?***
   ☐ *Yes*
   ☐ *No*
   ☐ *Unsure*

11. ***How important is it to you that machine availability updates instantly after a booking?***
    ☐ *Very important*
    ☐ *Somewhat important*
    ☐ *Not important*

Section D: Safety and Security

12. ***How should the system verify a user's identity before allowing a booking?***
    ☐ *Student ID number*
    ☐ *Hall email address*
    ☐ *Password and verification code*
    ☐ *Other (please specify):* ________________________

13. ***Are you concerned about non-residents using the system to reserve machines?***
    ☐ *Yes*
    ☐ *No*
    ☐ *Not sure*

14. ***How important is it to you that your personal and booking data remain private?***
    ☐ *Very important*
    ☐ *Somewhat important*
    ☐ *Not important*

15. ***Should the system automatically log users out after a period of inactivity?***
    ☐ *Yes*
    ☐ *No*
    ☐ *Not sure*

INTERVIEW

An interview consisting of 4 open-ended questions was conducted with Caudillo Jones, a Taylor Hall resident, on October 20, 2025.

1.What challenges do you face when trying to book or access a washing machine using the current system?

2.Have you ever experienced scheduling conflicts or long wait times when doing laundry? If yes, describe what happened.

3,How do you currently know when machines are free, and does this method cause confusion or delays?

4.What changes would make the current laundry process more organized and convenient for you?

Software Design Specification

for

Laundry System

Version 1.0

Prepared by

Ti-Jean Dehaney	*ID #620172199*	*tijean.dehaney@mymona.uwi.edu*
Darian Matthews	*ID #621069906*	*Darian.Matthews@mymona.uwi.edu*
Caudillo Jones	*ID #620172731*	*caudillo.jones@mymona.uwi.edu*
Ruth-Ann Allen	*ID # 620161125*	*ruthann.allen@mymona.uwi.edu*
Tristan Thompson	*ID #620153718*	*tristan.thompson02@mymona.uwi.edu*
Sean Thompson	*ID #620170129*	*sean.thompson02@mymona.uwi.edu*

Course Instructor:	**Mr. Ricardo Anderson**
Course:	***COMP2140 – Introduction to Software Engineering***
Studio Facilitator:	***Claudine Allen***

1.0 Project Overview

Taylor Hall currently relies on a disorganized, first-come-first-served laundry process, leading to overcrowding, long wait times, conflicts over machine usage, and inefficient resource management. The Laundry System is designed to modernize this process through a digital scheduling, ID verification, and notification system accessible through a web interface. The system allows residents to view available time slots, book appointments, reschedule or cancel bookings, and receive notifications. Hall staff can verify bookings, while administrators manage the entire schedule, generate reports, and resolve conflicts. The system improves convenience, reduces disputes, and ensures fair access to laundry facilities.

The intended users of the Laundry System include residents, who will book and manage their laundry appointments; laundry staff, who will verify bookings using student IDs or booking codes; and administrators, who will oversee system-wide bookings and generate reports. To ensure efficient and reliable operation, the system must meet several key non-functional requirements. These include maintaining a response time of under five seconds, providing real-time updates to prevent double bookings

2.0 Architectural Design

2.1 General Constraints

The Laundry System runs as a web application over the existing Taylor Hall and UWI network, which has limited bandwidth and unstable Wi-Fi in some areas. This constraint pushes the design toward light web pages, small request payloads, and simple real-time updates for bookings. The system must respond in under five seconds, so the design needs efficient database queries, minimal image use, and caching for common data such as time slot lists. Peak load also matters, since many residents try to book at similar times, so the design must support concurrent requests without long waits.

Hardware and software environments also place limits on design choices. Residents use a mix of low-end phones, laptops, and shared lab machines, so the system must run in standard browsers without plugins. Laundry staff and administrators often use shared office desktops with strict campus security policies, so the server side must use approved software stacks and standard ports. Budget constraints lead to a single shared application server and database, so the design must keep deployment simple and support monitoring and backup within that small footprint.

.

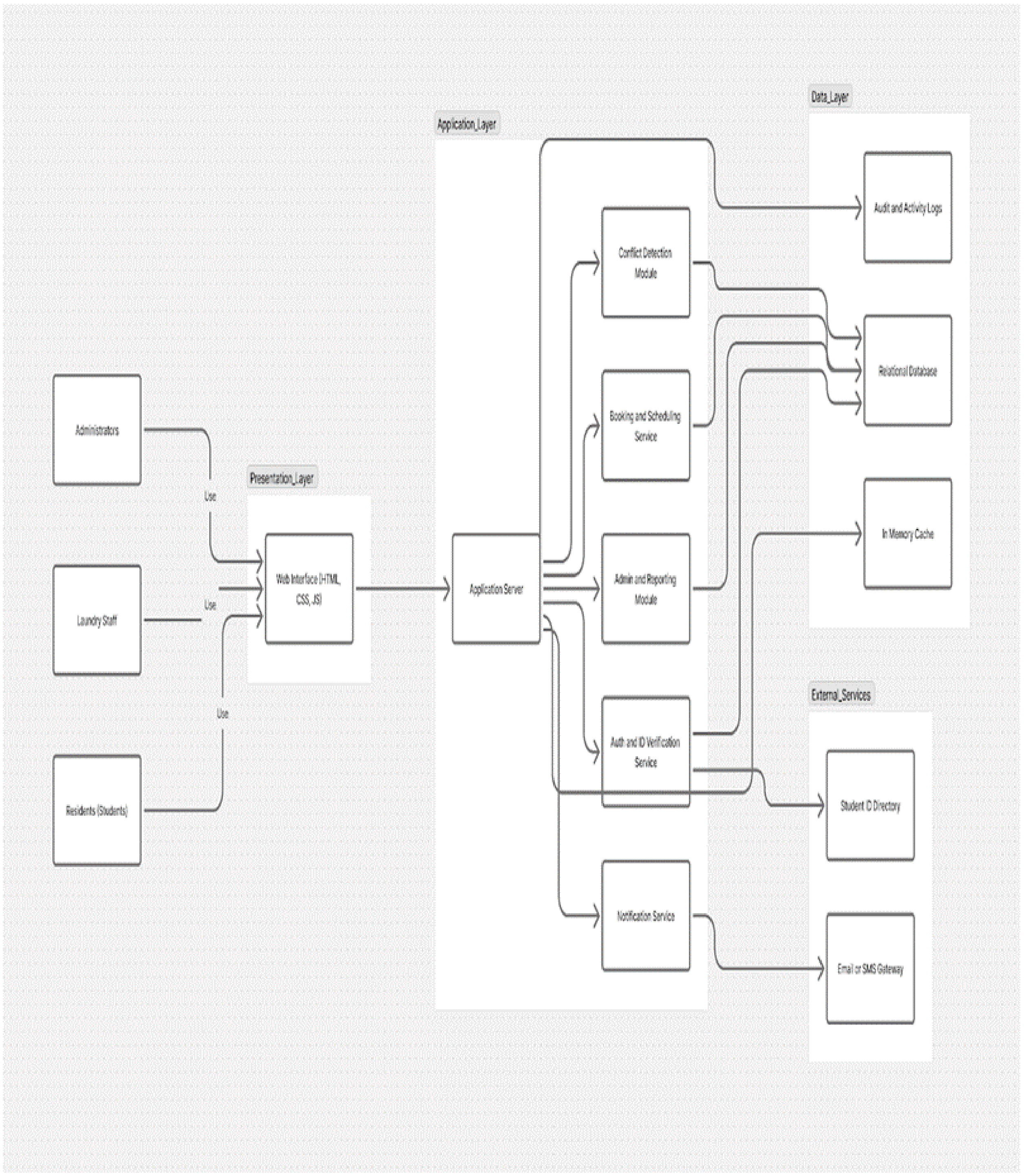
2.2 Alternatives Considered

The chosen architecture uses a three-tier, web-based, layered model. A thin browser client handles the user interface. An application layer on the server handles booking rules, ID checks, conflict resolution, and notifications. A database layer stores users, machines, and time slots. This pattern fits a campus web service, supports gradual growth, and keeps presentation, business logic, and data separate.

One alternative pattern is a classic two-tier client-server model with a thick desktop client. In that approach a desktop application on each machine performs most business logic and talks directly to the database server. This pattern offers strong performance for small, fixed user groups and can support richer offline features. For Taylor Hall, this pattern creates serious problems. Deployment and updates would become hard, because residents use many devices and do not control all of them. Security risk would rise, since many client machines would need direct database access. Cross-platform support would also suffer, since a desktop client would need separate versions for different operating systems.

A second alternative is a microservices architecture. In that pattern the team splits the system into many small services, for example, a booking service, notification service, user profile service, and reporting service, each with separate deployment and databases or schemas. This pattern suits large organizations with many teams, very high loads, and complex scaling needs. For Taylor Hall, the overhead would outweigh the benefits. The team would need to manage complex interservice communication, service discovery, and failure handling for a small student project. Monitoring, logging, and testing would become harder, while the functional scope remains modest. The three-tier layered model matches the size, skills, and deployment context of the Laundry System more closely than microservices or a thick client approach

2.3 System Architecture Diagram



2.3.1 Architectural Description

1. Client/User Components

1.1 Residents (Students)

Role in the system:
Residents are the main users of the laundry system. They:

- View available laundry time slots for specific machines and dates.
- Create new bookings.
- Reschedule or cancel existing bookings.
- View their upcoming and past appointments.
- Receive notifications for confirmations, reminders, and cancellations.

Interactions:

- Use a web browser (laptop, phone) to access the web interface.
- Send requests such as “view slots,” “book slot,” and “cancel booking,” which are forwarded to the application server.
- Receive real-time feedback if a slot becomes unavailable while they are choosing it, because the Booking & Scheduling Service validates slot availability against the database.

This directly supports the functional requirements of booking, managing, and viewing appointments.

1.2 Laundry Staff

Role in the system:

Laundry staff operate the physical laundry room and:

- Verify that a resident standing at a machine has a valid booking.
- Check bookings using student ID or booking code.
- Mark no-shows or report issues to administrators.

Interactions:

- Use their own login via the Web Interface with “staff” role.
- When a student arrives, staff enter the ID or booking code. The request is processed by:
    - Auth & ID Verification Service (check staff’s permissions).
    - Booking & Scheduling Service (retrieve booking, check time window, machine, and status).

- Staff see a simple “valid / not valid / wrong time / wrong machine” display.

This supports the functional requirement for ID verification and reducing conflicts in the laundry room.

1.3 Administrators

Role in the system:

Hall administrators supervise the whole operation. They:

- View the full schedule for all machines and days.
- Override or edit bookings if necessary.
- Resolve conflicts (double bookings, broken machines, misuse).
- Generate usage reports (peak hours, machine utilization, and user statistics).
- Configure rules (max bookings per week, allowed time windows).

Interactions:

- Use the Web Interface with the “admin” role.

- Their actions are routed to the Admin & Reporting Module and Conflict Detection Module, which talk to the database.

- Reports are generated and can be displayed on screen or exported.

This supports the functional requirements for schedule oversight, conflict resolution, and report generation.

2. Presentation Layer

Web Interface (HTML/CSS/JS, Browser UI)

Role in the system:

The Web Interface is the front-end that all user types see. It:

- Provides different views for residents, staff, and administrators based on role.

- Shows available slots in a calendar or timetable format.

- Shows booking forms, verification screens, and reports.

- Handles client-side validation and user experience.

Interactions:

- Sends HTTP requests (e.g., REST calls) to the application server.

- Receives JSON responses with time slots, booking status, and error messages.
- Can refresh the schedule regularly (e.g., polling) or use WebSockets (if implemented later) for near real-time updates.

The web interface is key to meeting usability and convenience requirements and contributes to performance by doing simple checks client-side before calling the server.

3. Application Layer Components

3.1 Application Server

Role in the system:

The Application Server hosts all backend services. It:

- Accepts incoming requests from the Web Interface.
- Routes them to the correct service:
    - Authentication, Booking, Notifications, Admin, Conflict Detection.
- Enforces security (only logged-in users can access protected operations).
- Coordinates database transactions.

Interactions:

- Calls underlying modules (Auth, Booking, etc.).
- Reads and writes to the database, reads from the cache, and writes to logs.
- Integrates with external systems like the Student ID Directory and Email/SMS Gateway.

It is the central “brain” ensuring all functional flows operate correctly and efficiently.

3.2 Authentication & ID Verification Service

Role in the system:

This service handles:

- User login (residents, staff, admins).
- Role-based access control.
- ID verification for bookings.

Interactions:

- On login, checks credentials against user records in the database.
- Determines user role (resident / staff / admin) and issues session tokens.
- When staff enter a student ID, it can:
    - Validate the ID against internal user records.
    - Optionally cross-check against an external Student ID Directory (if the hall or UWI central system is integrated).
- Ensures only authorized users can create bookings, verify bookings, or access admin features.

This supports security and fair access by preventing unauthorized use and fake bookings.

3.3 Booking & Scheduling Service

Role in the system:

This is the core logic for handling time slots and bookings. It:

- Manages the catalog of machines and time slots (e.g., 1-hour slots per machine).
- Creates new bookings if a slot is free.

- Reschedules or cancels existing bookings.
- Enforces rules:
    - No overlapping bookings for the same machine and time.
    - Limits per resident (e.g., max N bookings per week).
    - Time windows (e.g., bookings only valid for the current semester).

Interactions:

- Reads the list of machines, current bookings, and time slots from the Database.
- On booking:
    - Checks cache or DB for up-to-date availability.
    - Uses database transactions and constraints to ensure two residents are not given the same slot.
    - If successful, write the booking to the DB.

    - Triggers the Notification Service to send a confirmation.

- On reschedule or cancellation:

    - Validates rules, updates the booking, and triggers notifications.

This service directly satisfies functional requirements for viewing, booking, rescheduling, and cancelling and supports the non-functional requirement of preventing double bookings and real-time updates.

3.4 Conflict Detection Module

Role in the system:

This module focuses on detecting and handling scheduling conflicts. It:

- Scans for overlapping bookings on the same machine and slot.

- Identifies invalid bookings (e.g., a machine marked as out of service but still booked).

- Provides admin tools to resolve conflicts (cancel or move one of the bookings).

Interactions:

- Uses the database to read existing bookings and machine status.

- Runs checks:
  - At booking time (immediate conflict check).
  - Periodically or on demand for data consistency.
- Provides the Admin UI with lists of conflicts and suggested resolutions.
- Works closely with the Booking & Scheduling Service to apply corrections.

This contributes strongly to the fairness and reliability of the system by ensuring the schedule does not contain conflicting or impossible bookings.

3.5 Notification Service

Role in the system:

Handles all user messaging. It:

- Sends booking confirmations (email/SMS).
- Sends reminders before the booking time.
- Sends notifications for cancelled or changed bookings.

- Optionally sends warnings if a user is late or repeatedly misses bookings.

Interactions:

- Receives triggers from:
    - Booking & Scheduling Service (new, rescheduled, and cancelled bookings).
    - Admin & Reporting Module (mass notifications, e.g., “machine 3 is down today”).
- Formats messages with booking details (time, machine, location).
- Sends messages via the email/SMS gateway external service.

This supports convenience and communication and helps reduce no-shows and misunderstandings, which indirectly reduces conflicts in the laundry room.

3.6 Admin & Reporting Module

Role in the system:

This module provides admin functionality and reporting. It:

- Shows dashboards with today’s and weekly schedules.

- Let admins mark machines as unavailable, adjust time slot rules, or override bookings.
- Generates reports:
    - Number of bookings per day/week.
    - Peak usage times.
    - Most used machines.
    - No-show rates.

Interactions:

- Reads aggregated data from the database.
- May use precomputed summary tables or views to improve performance.
- Writes changes like updated machine status or admin overrides back to the DB.
- Can trigger the Notification Service when admin changes affect many users.

This supports the requirement for administrative oversight, reporting, and resource management improvement.

4. Data Layer Components

4.1 Relational Database

Role in the system:

Stores all persistent data, such as:

- Users (residents, staff, admins), roles, and login credentials (hashed).
- Machines (ID, name, status: available, broken, maintenance).
- Time slots (date, time, machine, capacity).
- Bookings (who, when, which machine, status).
- Audit logs and configuration.

Interactions:

- All application layer services read from and write to the DB.

- Uses constraints and indexes:
    - Unique constraint on (machine, timeslot) to enforce no double bookings.
    - Foreign keys linking bookings to users and machines.
- Transactions ensure consistency when multiple actions happen at once.

The DB is essential for correctness, consistency, and real-time availability checking, directly supporting the functional and non-functional requirements.

4.2 In-Memory Cache

Role in the system:

Improves performance and supports the less than 5 second response time requirement. It:

- Stores frequently accessed data such as:
    - Today’s and tomorrow’s available time slots.
    - Machine list and basic status.
    - Currently active bookings for the next few hours.

Interactions:

- The application server first checks the cache for read operations like “list available slots.”
- If data is in cache, the response is fast.
- When bookings change, the Booking & Scheduling Service updates the DB and invalidates or updates the relevant cache entries to keep data fresh.

By reducing repeated DB reads, the cache helps the system respond quickly, especially when many residents check availability at the same peak times.

4.3 Audit and Activity Logs

Role in the system:

Logs provide traceability and support dispute resolution. They:

- Record key actions:
    - Logins and logouts.
    - Booking creation, change, and cancellation.
    - Staff verification looks.

    - Admin overrides and conflict resolutions.

Interactions:

- The application server writes log entries for each important operation.

- Admins can review logs from the Admin & Reporting Module if there is a dispute (e.g., “I had a booking, and it disappeared”).

Logs support accountability, security auditing, and fair conflict resolution.

5. External Services

5.1 Student ID Directory

Role in the system:

This could be an internal UWI or Hall directory containing valid student IDs. It:

- Serves as an authoritative list to verify that a student ID is real and active.

Interactions:

- The Authentication & ID Verification Service may query this directory when:

    - A new resident account is created.

    - Staff scan or manually enter an ID in the laundry room.

- The system can reject bookings or verifications for IDs that are not known or not active.

This strengthens the security and integrity of the system, preventing non-residents from using the laundry facilities.

5.2 Email/SMS Gateway

Role in the system:

Delivers messages to phones or emails. It:

- Sends the actual text or email messages produced by the Notification Service.

Interactions:

- The Notification Service calls the gateway's API with the recipient's contact information and message content.

- The gateway handles sending, retries, and status.

This is essential for the notification feature and improves user experience and punctuality.

6. How Components Interact in Key Flows

6.1 Resident Books a Time Slot

1. The resident logs in via the web interface.
2. The web interface sends the login to the application server, which calls the auth service.
3. After authentication, the resident opens the “Schedule” page.
4. The web interface requests available slots from the application server.
5. The application server checks the cache; if not present, it queries the booking & scheduling service, which reads from the database and populates the cache.
6. The resident chooses a slot and submits.
7. The application server calls the booking & scheduling service to create a booking:
    - Starts a DB transaction.
    - Checks if the slot is still free (no existing row for that machine/slot).
    - If free, inserts a booking row.

    - Commits the transaction, ensuring no double booking.

8. Booking & Scheduling Service triggers Notification Service.

9. The notification service sends confirmation via email/SMS gateway.

10. A log entry is written to audit logs.

6.2 Staff Verifies Booking

1. The resident arrives with a student ID or booking code.

2. Staff uses the web interface to enter ID or code.

3. The request goes to the application server.

4. Auth Service confirms staff role.

5. The booking & scheduling service retrieves any active booking matching ID and current time and machine.

6. If valid, the system shows “Approved” to staff; otherwise, it shows the appropriate error.

7. Action is logged.

6.3 Admin Resolves Conflict and Generates Report

1. Admin logs in via web interface.
2. Opens the “Conflicts” dashboard.
3. The Admin & Reporting Module queries the Conflict Detection Module.
4. The Conflict Detection Module scans bookings and marks conflict cases.
5. Admin selects which booking to move or cancel.
6. Booking & Scheduling Service updates records, and Notification Service informs affected residents.
7. Admin later runs a monthly report; the Admin & Reporting Module queries aggregated statistics from the database and displays them.

7. How the Architecture Meets Requirements

Functional Requirements

- View available time slots:
  Web Interface + Booking & Scheduling Service + DB + Cache.
- Book, reschedule, or cancel:
  Booking & Scheduling Service with transactional DB operations ensures consistent bookings.
- ID verification by staff:
  Auth & ID Verification Service + Student ID Directory + Booking & Scheduling Service.
- Admin oversight and reporting:
  Admin & Reporting Module + Conflict Detection Module + DB + Logs.
- Notifications:
  Notification Service + Email/SMS Gateway.

Non-Functional Requirements

- Response time less than 5 seconds:
  - In-memory cache for frequently accessed schedule data.
  - Optimized queries and indexes in the relational database.
  - Clear separation of layers (presentation, application, and data) so the system can be tuned and scaled if needed.

    - Minimal data transfer by returning only necessary information.

- Real-time updates/no double bookings:
    - Single central booking & scheduling service with strict DB constraints on machine/slot combinations.
    - Immediate availability checks during booking.
    - Cache invalidation or refresh on changes to keep displayed availability accurate.
    - Optional short polling or live updates from the server to refresh the schedule on user screens.

- Security and fair access:
    - Auth & ID Verification Service enforces logins and role-based access.
    - Integration with the Student ID Directory ensures only valid residents use the system.
    - Booking rules (limits per week, etc.) are enforced in Booking & Scheduling.

    - Logs support investigation and discourage abuse.

- Reliability and dispute resolution:
    - All changes are stored in the DB with timestamps.
    - Audit logs record who did what and when.
    - Admin tools to resolve conflicts and view history.

- Maintainability and extensibility:
    - Modular services: Auth, Booking, Notifications, Admin, and Conflict Detection.
    - New features (e.g., mobile app, QR code verification at machines) can be added by reusing the same application layer and data layer.

2.3.2 Architecture Justification

The architecture uses separate layers so each part stays focused on its role. The interface handles user actions. The services handle booking rules, verification, notifications, and admin work. The database stores all state. This structure keeps the system organized, cuts down errors, and avoids mixing logic with interface code. It also makes updates easier because each part stays isolated.

The architecture meets the performance and accuracy needs of the hall. The booking service runs all checks through controlled transactions. The database holds strict rules so two residents never take the same slot. The cache speeds up heavy tasks like viewing the schedule so the system stays under the five-second response time target. The authentication service protects access, so only residents, staff, and admins use the system in the right way.

The architecture stays flexible because each service stands on its own. New features such as QR check-in or new alerts fit in without changing how core booking works. Admin work, reporting, and conflict checks run in their own modules so they do not slow down residents. This structure fits your hall because it stays simple to manage while still giving room for growth.

3.0 Architecture Decomposition

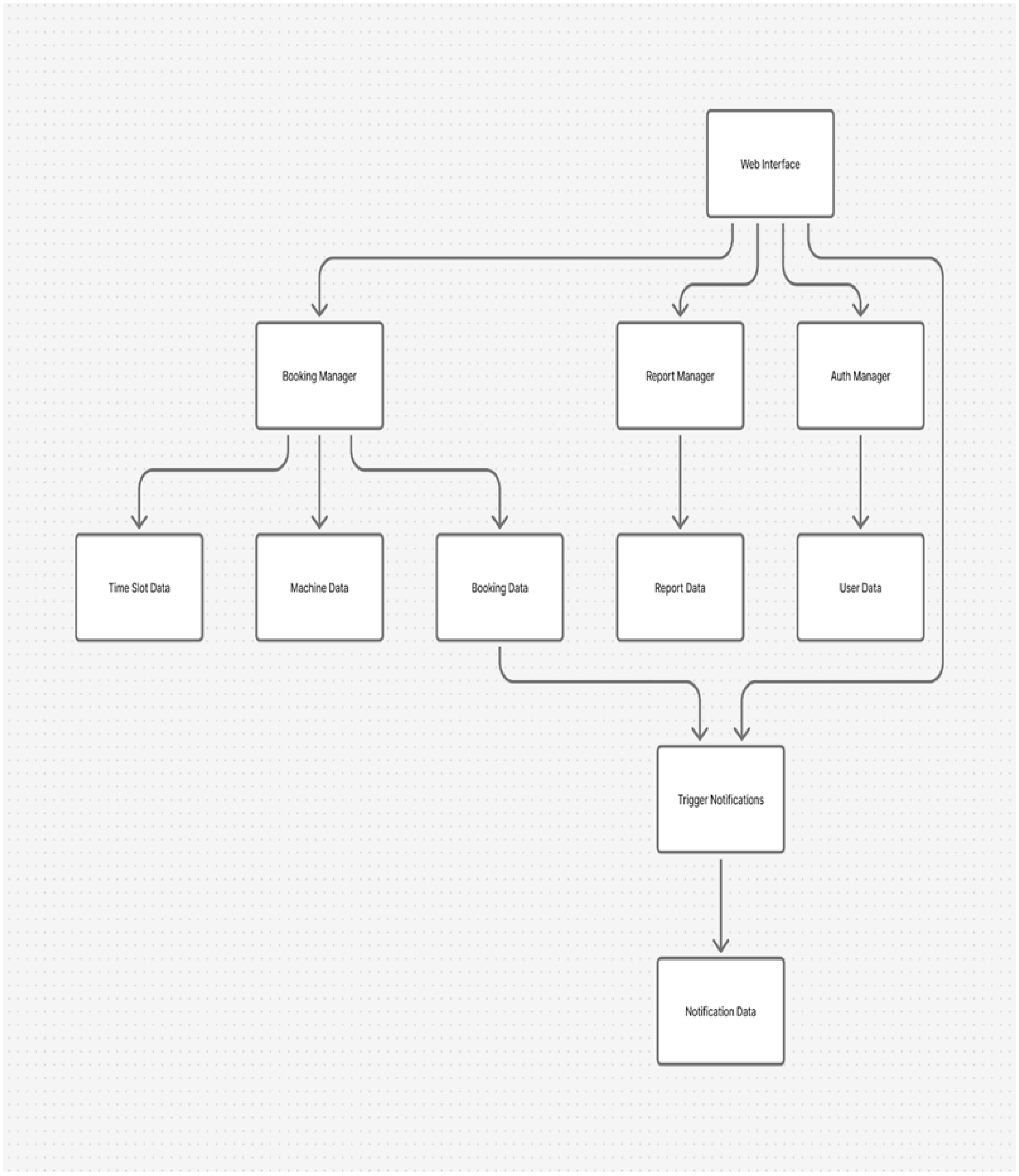
3.1 Component Decomposition – Modules

Requirement ID	Architecture Component	Module Name	Description

Req. 1 Req. 3 Req. 6.1	User Interface	SystemUI SystemTXT	This will provide the layout and attendant functions for enabling the user to manage access to the various features when the system starts up. This will provide a minimal text-based interface that will allow system setup, to create the initial system users and modify administrative privileges.

Req. 2.3 Req 1.4 Safety & Security	Authorization	AdminAuth PrivilegeSys	Provides authority flags and revocation to different levels of users Interact with components seeking to verify a users' identity

3.2 Structural Design – Structure Chart



3.2.1 Design Notes

The structure chart groups the system into four main modules: the Auth Manager, Booking Manager, Notification Manager, and Report Manager. These modules handle all system logic, while the web interface acts as the entry point for residents, staff, and administrators. Each module connects only to the data it needs, which keeps the system simple and reduces the risk of errors. The Auth Manager interacts with the User Data store to check login details and roles. The Booking Manager handles all booking tasks and connects to the booking, machine, and timeslot data. The Notification Manager creates and logs messages linked to bookings. The report manager reads historical booking data and writes reports for administrative use.

The relationships shown are intentional. The web interface sends all actions to the managers because this keeps business rules out of the UI. The Booking Manager links to the Notification Manager through the booking store, because notifications depend on booking actions such as creating or rescheduling. No module writes directly to another module’s internal data, which keeps responsibilities clean. This structure supports strong control over booking constraints, such as preventing double-bookings and enforcing time windows.

The design assumes that all persistent data is stored in separate repositories (users, bookings, machines, time slots, notifications, and reports). Each manager only reads or writes what it needs, which avoids mixing data and reduces complexity. The structure chart uses one-directional connections to show that the web interface triggers managers, not the other way around. This fits the workflow of the laundry system and keeps the logic easy to maintain and extend.

Test plan

TEST	TEST DATA	Associated requirement	Expected Result	Actual Result	Pass/Fail
Case 1: View Available Time Slots “Student login with email or ID number and password.”	“Student login with email or ID number and password.”	Req #1	Displays available time slots	Displayed available time slots	Pass
Case 2: Book Laundry Appointment “The student selects a machine and dryer from the available list and it doesn't allow double booking per person. ”	“Washer one and dryer 1.” At 9 am.	Req #2	Displays a confirmation message if the dryer is available.	Displayed a confirmation message as the dryer is avaliable	Pass
Case 3: Reschedule Appointment “The student goes to my bookings section and selects the reschedule machine option.”	“Washer 2, dryer 2” “Time: 11 am for both”	Req #3	Allows the user to select the reschedule option in the my booking section and a pop-up comes up	The user was able to reschedule their appointment.	Pass

			that allows the user to select a new date, time and washer.		
Case 4: Cancel Appointment	"The student goes to my bookings section and selects the cancel option."	Req #4	Displays a success message that states the appointment has been cancelled.	Displayed a success message.	Pass
Case 5: Receive Booking Confirmation	"Once a machine is selected, the user will receive email verification and notification."	Req #5	Once the booking is made, a success message is displayed and an email is received.	Message received	Pass
Case 6: Administrator Dashboard The admin who logs in is given access to the dashboard And are able to see current appointments."	Remove Sean's appointment.	Req #6	Displays a message asking for confirmation	Appointmentnt removed.	Pass
Case 7: Admin Manage all bookings	"Add a new machine and dryer."	Req #7	Displays a success message that says machines added.	Success message was displayed and the new	Pass

“The admin is able to manage all the bookings and remove or add machines.”				machines were added	
Case 8: Generate user report	“Admin Select generate report”	Req #8	Downloads an Excel file with all the students, the washers and dryers they used, and the date and time.	Downloaded an Excel file and displayed all the students, the washers and dryers they used, and the date and time.	Pass

Github link:
https://github.com/Tristan-Thompson876/Taylor-Laundry-System.git