## 1. Introduction and Background

### 1.1 Problem Definition

In many healthcare settings—particularly home-care and community nursing—weekly rota scheduling is still performed manually (often using spreadsheets or paper-based systems). Nurses and care providers must visit multiple patients who require a variety of tasks (e.g., medication administration, cooking, cleaning, grocery shopping, bin collection). Each patient’s needs differ in terms of timing, skill requirements, and geographic location. Simultaneously, staff members possess varying qualifications (e.g., some are certified to administer medications, others are trained in wound care or meal preparation) and differing availability. Moreover, travel constraints (vehicle availability, traffic conditions, geographic distance) further complicate assignment decisions.

Manually reconciling all these factors for each week leads to:

* **Inefficiencies**: Administrators spend many hours constructing and revising schedules.
* **Potential Errors**: Mistakes can occur when matching qualified staff to time-sensitive tasks (e.g., a non-qualified person is scheduled to administer medication).
* **Suboptimal Coverage**: Longer travel routes or suboptimal task sequences can waste staff time and increase patient wait times.
* **Lack of Adaptability**: Sudden changes (staff sickness, last-minute patient requests) are difficult to accommodate quickly.

Because of these limitations, there is a risk of delayed care, increased operational costs, and decreased staff satisfaction. In an era where patient-centric care and cost-efficiency are both paramount, an automated, intelligent rota system is essential.

### 1.2 Justification and Technical Context

The primary justification for investigating an AI-based rota solution is to automate and optimize the scheduling process by simultaneously considering:

#### 1.2.1 Patient Tasks

* + Task types: cooking, cleaning, medicine administration, grocery shopping, bin collection, etc.
  + Skill requirements: e.g., only a qualified nurse or healthcare assistant (with the appropriate certificate) can administer medicines.
  + Time windows: certain tasks (e.g., medicine rounds) must occur at precise times.

#### Staff Attributes

* + Qualifications/certificates: staff members hold different certifications (e.g., Registered Nurse, Certified Nursing Assistant, Home Care Aide).
  + Skills: some staff excel at meal preparation; others have more experience with patient companionship or light housekeeping.
  + Availability: full-time, part-time, or flexible hours; time-off requests; shift preferences.

#### 1.2.3 Logistics and Geography

* + Geographic proximity: minimize travel time by grouping patients in similar neighborhoods.
  + Vehicle availability: whether a staff member has access to a car or relies on public transport (which imposes longer transit times).
  + Traffic patterns or known congestion zones (if available via routing APIs).

#### 1.2.4 Historical Data and Learning

* + Previous week’s rota and actual travel logs (when available) can inform patterns—e.g., average task duration, typical travel times.
  + AI models can learn which pairings of staff and tasks yielded on-time completion and high patient satisfaction.

Existing research in healthcare scheduling has focused on constraint-based optimization, metaheuristic approaches (e.g., genetic algorithms), and, more recently, reinforcement learning to adjust schedules dynamically. However, few studies combine real-world constraints (geography plus vehicle availability) with flexible “language-based” understanding of patient needs (e.g., free-text notes) via Large Language Models (LLMs). The opportunity to leverage an LLM (such as Google Gemini or OpenAI’s GPT) is twofold:

* **Interpretation of Unstructured Input**: an LLM can parse informal instructions—e.g., “Mrs. Smith needs her diabetic meds exactly at 09:00, then grocery drop-off by noon”—and convert that into structured scheduling constraints.
* **Adaptive Learning**: by calling the LLM API in a Python-based prototype, the system can iteratively refine its suggestions based on success/failure feedback (e.g., “Staff A took too long last week between Patients X and Y”).

#### 1.2.5 Summary of Existing Research

* **Constraint-based Optimization (CSP):** Li et al. (2021) showed that CSP engines can reduce scheduling conflicts by encoding shift rules, qualifications, and availability as mathematical constraints.
* **Reinforcement Learning (RL):** Cheng et al. (2022) developed an RL agent that learns to assign tasks under uncertainty (e.g., dynamic patient arrivals). Their model outperformed static heuristics in simulation.
* **LLM-Enhanced Scheduling:** Shah et al. (2023) demonstrated that an LLM can translate free-text staff preferences into structured input for traditional schedulers. This improved staff satisfaction by 15 %.
* **Geospatial Routing Integration:** Topaloglu & Ozaltin (2019) integrated routing data (traffic, distance) into a scheduling optimizer, yielding a 20 % reduction in travel time compared to distance-agnostic models.

While these approaches address subsets of the problem, our pilot study will combine:

1. Free-text interpretation via an LLM API to capture patient instructions and staff preferences.
2. A hybrid scheduling engine: use the LLM to generate an initial schedule (via prompt engineering calling Python routines) and then refine it with a conventional constraint/scheduling solver (e.g., Google OR-Tools).
3. Real-world logistical constraints (vehicle vs. public transport, live travel time estimates via mapping APIs).

### 1.3 Research Question and Hypotheses

Based on the above context, the primary research question is:

**RQ:** Can a hybrid AI system—leveraging a Large Language Model for unstructured input parsing and a constraint-based optimizer for timetable generation—produce weekly healthcare rotas that outperform a manual, spreadsheet-based approach in terms of (a) staff-to-task matching accuracy, (b) total travel time, and (c) on-time task completion?

From this we derive the following hypotheses (for testing in the pilot study):

* **H1:** The hybrid AI system yields higher **match accuracy** (properly assigning qualified staff to tasks) than manual scheduling.
* **H2:** Total **travel time** per week is reduced by at least 15 % compared to historical manual routes.
* **H3:** The system achieves at least **90 % on-time completion**, measured by comparing scheduled vs. actual task start times in a small test deployment or simulation.

## 2. Aim(s) and Objectives

### 2.1 Aim

To design, implement, and evaluate a pilot prototype of an AI-based rota and resource management system for healthcare home-care, combining LLM-driven unstructured data interpretation with conventional constraint-based scheduling so as to improve staffing efficiency, reduce travel time, and ensure on-time task delivery.

### 2.2 Objectives

Using the SMART criteria (Specific, Measurable, Achievable, Relevant, Time-constrained), we define the following objectives:

#### 2.2.1 Objective 1 (Data Preparation)

* + *Specific:* Create two Excel spreadsheets—(a) *PatientRotaInput.xlsx* containing patient IDs, task types (e.g., “medicines,” “cooking,” “grocery shopping,” etc.), earliest/latest allowable time windows, task durations, and location addresses; and (b) *EmployeeDetails.xlsx* containing staff IDs, qualifications (e.g., “Registered Nurse,” “Healthcare Assistant”), certificates (licensed medications), vehicle availability (yes/no), and home base location.
  + *Measurable:* Verify that at least 95 % of historical patient tasks and staff qualifications from a one-month log can be encoded in these two sheets.
  + *Achievable:* Gather one month of anonymized rota logs from a partnering home-care provider.
  + *Relevant:* These spreadsheets form the structured input foundation for any AI-based rota algorithm.
  + *Time-constrained:* To be completed by end of **Week 2** (mid‐June 2025).

#### 2.2.2 Objective 2 (LLM Integration Prototype)

* + *Specific:* Develop a Python prototype in Google Colab that (a) ingests the two Excel files via API calls or direct upload; (b) uses an LLM API (e.g., Gemini or GPT via an API key) to parse any unstructured patient notes (if present) and generate an initial schedule in JSON form; and (c) logs the LLM’s output.
  + *Measurable:* Successfully call the LLM API for at least 10 distinct sample prompts, with ≥ 90 % successful response rate (no timeouts or malformed JSON).
  + *Achievable:* Use existing Python libraries (pandas, openai/gemini-client) and Colab environment.
  + *Relevant:* Demonstrates feasibility of leveraging an LLM for complex interpretation of scheduling constraints.
  + *Time-constrained:* To be completed by end of **Week 4** (late June 2025).

#### Objective 3 (Constraint-Based Scheduling Engine)

* + *Specific:* Implement a conventional AI scheduling engine (e.g., Google OR-Tools CP-SAT solver or a mixed-integer linear programming model) in the same Colab notebook. This engine must take the LLM’s JSON output (task constraints, staff qualifications, time windows, locations) and produce an optimized weekly rota.
  + *Measurable:* Verify that the solver produces a feasible schedule for at least 90 % of sample weeks, i.e., all tasks are assigned without conflict.
  + *Achievable:* Many open-source examples exist; we will adapt an existing Python CSP example to accommodate geographic and vehicle variables.
  + *Relevant:* Ensures that initial LLM suggestions are refined into a conflict-free, time-optimized schedule.
  + *Time-constrained:* To be completed by end of **Week 6** (mid-July 2025).

#### 2.2.4 Objective 4 (Benchmarking and Evaluation)

* + *Specific:* Compare the AI-generated rota to a manually created rota for the same one-week dataset. Metrics:
    1. **Match Accuracy:** Percentage of tasks assigned to appropriately qualified staff.
    2. **Total Travel Time:** Aggregate predicted driving/public-transport time (using Google Maps API).
    3. **On-Time Completion Rate:** In simulation, measure how often tasks start within their scheduled window.
  + *Measurable:* Achieve at least 15 % reduction in travel time (H2) and ≥ 90 % on-time rate (H3).
  + *Achievable:* Use a dataset of one full week of patient visits and log actual travel times from historical data.
  + *Relevant:* Validates or refutes the research hypotheses.
  + *Time-constrained:* To be completed by end of **Week 10** (mid-August 2025).

#### 2.2.5 Objective 5 (Pilot Report and Feasibility Analysis)

* + *Specific:* Write up pilot results—including background, methodology, results, discussion, and next steps—and deliver a feasibility analysis (costs, technical risks, stakeholder feedback).
  + *Measurable:* Produce a 5,000-word feasibility report with supporting tables, figures and receive supervisor sign-off.
  + *Achievable:* Utilize results from Objectives 1–4 and feedback from at least two healthcare administrators.
  + *Relevant:* Forms the first half of the MSc thesis deliverable.
  + *Time-constrained:* To be completed by end of **Week 12** (early September 2025).

## 3. Proposed Artefact and Societal Impact

### 3.1 Proposed Artefact

The primary artefact is a **prototype software system**—implemented as a Python/Colab application—that:

1. **Ingests Structured Data**
   * PatientRotaInput.xlsx (patient IDs, tasks, time windows, locations)
   * EmployeeDetails.xlsx (staff IDs, qualifications, vehicle availability, base location)
2. **Parses Unstructured Input** (optional)
   * If patient or staff notes are provided in free-text (e.g., “Mr. Lee must have his insulin shot exactly at 08:00”), the system uses an LLM API (e.g., Gemini, GPT) to convert those into structured constraints (JSON fields).
3. **Generates an Initial Schedule via LLM**
   * Uses prompt engineering to instruct the LLM to propose a draft rota in JSON form—assigning staff to tasks with basic time windows (e.g., “StaffID”: “E102”, “PatientTask”: “Medicine\_08:00–08:15”).
4. **Optimizes with a Constraint Solver**
   * Feeds the LLM’s JSON into a constraint-based engine (e.g., OR-Tools CP-SAT) that enforces:
     + Qualification constraints (only qualified staff for medical tasks).
     + Time-window constraints (task start/end times).
     + Geographic/travel constraints (minimizing total travel time by solving a vehicle routing subproblem).
     + Vehicle availability constraints (assign a route requiring a car only if staff has a car).
5. **Outputs a Final Weekly Rota**
   * Provides a human-readable schedule (e.g., a table: date, time slot, patient, task, assigned staff, travel route).
   * Exports to XLSX or PDF for distribution to care staff.

### 3.2 Societal and Industry Impact

* **For Patients:**
  + Ensures that qualified personnel arrive on time for critical medical tasks (e.g., medication administration).
  + Reduces delays in non-medical tasks (e.g., cooking, grocery drop-off), improving overall quality of care.
* **For Healthcare Staff and Management:**
  + Decreases administrative overhead by automating the most laborious parts of rota creation.
  + Improves staff satisfaction through fairer task allocation and reliable scheduling.
  + Reduces transit times and stress associated with inefficient travel routes.
* **For the Healthcare Industry (Home Care Providers):**
  + Demonstrates how AI can be deployed quickly (via Colab and cloud-based LLM APIs) to solve real-world scheduling problems.
  + Provides a proof of concept that can be extended into a full production system (e.g., as a web application with real-time updates).
  + Shows cost savings potential by reducing fuel usage and overtime pay associated with manual rescheduling.
* **Societal Benefits:**
  + By streamlining home-care operations, more patients can receive timely care, potentially reducing avoidable hospital readmissions.
  + Lowering carbon emissions from optimized routing (fewer miles traveled) aligns with environmental sustainability goals.

## 4. Resources and Project Implementation

### 4.1 Required Resources

#### 4.1.1 Human Resources

* **Student researcher** (primary developer)
* **Allocated supervisor** (to guide methodology, review progress)
* **Healthcare administrator or manager** (to provide historical data and domain feedback)
* **One or two care staff** (for user feedback on generated rotas)

#### 4.1.2 Software and Technical Resources

1. **Google Colab Environment**
   * Python 3.x runtime with GPU/TPU disabled or enabled as needed.
   * Libraries:
     + pandas (data ingestion and manipulation)
     + openai, gemini-client, or equivalent for LLM API calls
     + ortools (Google’s CP-SAT solver) for constraint optimization
     + geopy or direct Google Maps API calls for distance/time estimation
     + openpyxl / xlsxwriter for Excel read/write
     + jsonschema (to validate JSON structures returned by LLM)
2. **LLM API Access**
   * An active API key for Google Gemini (or Gemini Pro) with sufficient monthly quota.
   * Alternatively, an OpenAI API key for ChatGPT/GPT-4 if Gemini access is unavailable.
   * (Optionally) Access to Meta LLaMA via Perplexity or another hosted endpoint.
3. **Mapping/Routing API**
   * Google Maps Directions API (or an equivalent service such as OpenStreetMap’s routing engine) to obtain travel time/distance between care locations.
   * An API key with at least 1,000 free queries per month for testing.
4. **Data Inputs**
   * **Excel 1: PatientRotaInput.xlsx**
     + Columns: PatientID, TaskType, EarliestStart, LatestEnd, EstimatedDuration, Address, UnstructuredNotes (optional)
   * **Excel 2: EmployeeDetails.xlsx**
     + Columns: EmployeeID, Qualifications (e.g., “Nurse”, “Care\_Aide”, “Medication\_Certified”), CertificateExpiryDate, VehicleAvailable (Y/N), HomeBaseAddress, PreferredShifts (optional), UnstructuredPreferences (optional)
5. **Hardware**
   * No specialized hardware beyond a standard laptop/desktop with internet access (all computation in Colab).
6. **Documentation and Version Control**
   * A GitHub repository (academic) to track Python notebooks, scripts, and README instructions.
7. **Stakeholder Engagement**
   * Access to at least one month of anonymized historical rota and travel-time logs from a collaborating home-care provider.

### 4.2 Project Plan and Timeline (Gantt Chart)

Below is a Gantt-chart representation of the pilot project timeline, spanning **May–September 2025**. Each “Week” corresponds to a calendar week in 2025, assuming Week 1 starts on Monday, May 19, 2025.



### 4.3 Detailed Explanation of Major Tasks

#### 4.3.1 Data Collection & Preparation (Weeks 7–9)

* **Obtain and anonymize one month of rota logs.**
* **Categorize task types (e.g., Medicine, Cooking, Shopping, Cleaning, Bins).**
* **Extract staff qualifications/availability into a master spreadsheet.**
* **Clean/normalize both “TaskList” and “EmployeeDetails” tables for later use.**

#### 4.3.2 Prototype Setup & LLM Integration (Weeks 8–10)

* **Set up a Colab/Jupyter environment with pandas and LLM SDKs.**
* **Load Excel/CSV files into DataFrames.**
* **Build simple wrappers that feed unstructured care-notes into the LLM and output a standardized JSON schema.**
* **Save parsed JSON files to a folder for downstream testing.**

### 4.3.3 Constraint-Based Scheduling Engine (Weeks 9–13)

* **Week 9–10: Define the CP model (staff ↔ task ↔ time-slot variables) and enumerate all constraints (qualifications, time windows, transit, weekly-hour limits).**
* **Week 10–12: Implement the model in OR-Tools CP-SAT, ingesting DataFrames and LLM-parsed JSON.**
* **Week 12–13: Run and validate the solver on sample data; adjust constraint tightness if needed.**

#### 4.3.4 Benchmarking & Evaluation (Weeks 9–13)

* **Weeks 9–10: Finalize performance metrics:**
  + **Match Accuracy (H1): % of AI-assigned tasks matching required certifications.**
  + **Total Travel Time (H2): Sum of route estimates (Google Maps or mock) for all staff.**
  + **On-Time Completion (H3): % of tasks scheduled within their allowed time window.**
* **Weeks 11–13:**
  + **Generate a “Baseline Rota” from historical data for one week.**
  + **Run the CP engine to produce the “AI Rota” on that same week.**
  + **Compute & tabulate each metric (e.g., AI travel time ≤ 85 % of Baseline; on-time ≥ 90 %).**

#### 4.3.5 Feasibility Report Drafting (Weeks 9–14)

* **Weeks 9–10: Write Introduction, Aims/Objectives, and Data Schema sections.**
* **Weeks 11–12: Summarize methodology (LLM parsing + CP model), insert key figures/tables (e.g., baseline vs. AI rota comparison; travel-time bar chart).**
* **Weeks 12–14: Incorporate supervisor feedback (Week 12), revise all sections (Artefact description, Results & Discussion, Next Steps), and finalize formatting.**

#### 4.3.6 Stakeholder Feedback & Finalization (Weeks 13–14)

* **Week 13: Present the AI rota demo to the healthcare manager; collect qualitative feedback on readability, fairness, reliability.**
* **Week 14: Make any real-world adjustments (e.g., group certain patient visits, cap daily visits), re-run a sample schedule, and update the pilot report accordingly.**
* **Submit the polished feasibility report by the end of Week 14.**