Abdullah Elmawy

Nurse Scheduling Problem Implementation

[Analysis 3](#_Toc176303739)

[Description of the Problem 3](#_Toc176303740)

[How Is the Problem Suitable to be Solved by Computational Methods? 3](#_Toc176303741)

[Description Of Stakeholders 5](#_Toc176303742)

[Investigating Existing Solutions 7](#_Toc176303743)

[Questionnaire Results 10](#_Toc176303744)

[Nurse 1: 10](#_Toc176303745)

[Nurse 2: 10](#_Toc176303746)

[Admin 1: 11](#_Toc176303747)

[Questionnaire Analysis 11](#_Toc176303748)

[Trends in Nurses 11](#_Toc176303749)

[Trends in Admin 11](#_Toc176303750)

[Features of the proposed solution 11](#_Toc176303751)

[Limitations Of Proposed Solution 13](#_Toc176303752)

[Hardware And Software Requirements 15](#_Toc176303753)

[Measurable Success Criteria 15](#_Toc176303754)

[Design 18](#_Toc176303755)

[Algorithms 18](#_Toc176303756)

[Genetic Algorithm 18](#_Toc176303757)

[Pseudo code: 18](#_Toc176303758)

[The supporting algorithms 20](#_Toc176303759)

[Greedy Initialisation Algorithm 20](#_Toc176303760)

[Pseudocode: 21](#_Toc176303761)

[Local Search Algorithm 22](#_Toc176303762)

[Psudocode: 22](#_Toc176303763)

[Simulated Annealing (SA) Algorithm 23](#_Toc176303764)

[Pseudocode: 23](#_Toc176303765)

[Constraint Propagation Algorithm 24](#_Toc176303766)

[Pseudocode: 24](#_Toc176303767)

[Usability Features 25](#_Toc176303768)

[Main Screens 25](#_Toc176303769)

[Key Data Structures to be Used and Design of the Database: 29](#_Toc176303770)

[Data Structures 29](#_Toc176303771)

[Database 29](#_Toc176303772)

[Test data 31](#_Toc176303773)

[The main objectives of the system and how the testing process addresses them? 31](#_Toc176303774)

[The main tests to test the robustness of the system 31](#_Toc176303775)

[Exemplar nurse data to be used 33](#_Toc176303776)

[Top-down Decomposition. 0](#_Toc176303777)

[0](#_Toc176303778)

# Analysis

Description of the Problem

Staff scheduling is a problem encountered in many organisations, such as call centres, educational institutions, industry, hospitals, and other public services. Staff scheduling is especially important to maximise efficiency while not violating the staff's personal needs.

The nurse scheduling problem (NSP) is a variant of staff scheduling problems which appoints nurses to shifts taking both hard constraints, i.e., legal requirements, and soft constraints, i.e., nurse’s preferences, into account. Thus, the timetable is designed to maximise the nurses' preferences and avoid violating any of the hard constraints. Various models and techniques have been proposed to address the NSP, ranging from simple to advanced approaches. [[1]](#footnote-2)

Different countries have different laws, and each hospital has its internal rules, the program should consider this and be flexible to adapt to changes set by staff.

## How Is the Problem Suitable to be Solved by Computational Methods?

Each hospital in the UK has its way of Scheduling nurses to shifts, but this makes the process dependent on the hospital and can lead to bad allocation of resources, thus costing the NHS. In other countries, nurses are allocated by hand, leading to many troubles. Allocating by hand can lead to over-coverage or under-coverage. Assigning by hand can also introduce bias, because of the overreliance on a human factor.

|  |  |  |
| --- | --- | --- |
| Feature making it suitable to use computational method | Challenge | Solution |
| Complex Constraints and Large Solution Possibilities | The problem involves managing many constraints, such as nurse preferences, legal regulations, skill requirements, and hospital policies. The possible combinations of shifts, days off, and preferences create a large solution space, making manual exploration impractical. | Computational algorithms can efficiently explore a vast number of combinations to find feasible and optimised schedules. The effort and time required to solve as accurately will be too much for any human, especially in large teams and organisations. Genetic algorithms are designed to explore large solution spaces and reach as good of a solution as feasibly possible in the time given, providing practical means to generate and evaluate numerous nurse schedules. |
| Changing Healthcare Environment | Healthcare environments are dynamic, with varying patient loads, staff availability, and unforeseen events, requiring adaptable scheduling solutions. For example, one nurse can be sick on the day or be off for training, these force the schedule to adapt, or the hospital will have to be understaffed for the day, offering a reduced standard of care to the patients. | Computational methods can learn from historical data and adapt to changing patterns, ensuring schedules remain efficient and responsive to dynamic conditions. An algorithm can also be more accurate with the substitutions made to cover any gaps in the schedules, as it will evaluate how one change will affect the schedule in the future thus minimising any complications resulting from the one change to the schedule. |
| Optimisation Goals and cost reduction | The goal is not just to create schedules that meet constraints but to optimise for factors like cost, staff satisfaction, and operational efficiency. This will help reduce the costs of the NHS saving more money to be spent on staff wages and refurbishments to hospitals, all helping to offer a higher standard of care to the patients. | Optimisation algorithms, such as genetic algorithms, excel in finding solutions that optimise specific criteria, allowing for fine-tuning of schedules, offering more than just a schedule that works but a schedule that benefits the hospital in the long term and the NHS overall. |
| Efficiency | Manually creating and managing nurse schedules is time-consuming, prone to errors, and may not be feasible as the complexity of the problem increases. This wastes valuable time of clinicians or administrators that could be better spent on the patients or fulfilling staff requests. | The algorithm can do in minutes what a human will do in multiple hours, the algorithm will also produce a more accurate result than the human trying to schedule it. The time wasted on producing the schedule can be massively cut if software is producing it, the software can also offer a nicer way for the staff to see their schedule or to raise complaints. This would save time for the administrator from having to follow up on every request. |

## Description Of Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| Stakeholder | Description of their role in a hospital | Their interest in the solution | How they will use the software |
| Nurses | Nurses work as part of a multidisciplinary team, providing direct patient care. Nurses work in a variety of settings, from hospital wards and operating theatres to schools and patients' homes. + | Nurses would love to be able to access their rota remotely and in an easy-to-read way. Nurses would also love to be able to request holidays, study leaves, and sick days without having to contact the department secretary personally, or worrying about harming patients because the hospital would be understaffed. Nurses would be interested in the solution as it lowers their stress levels because they do not need to worry about the rota, but only about doing the job they are trained to do. | Nurses will have the software on their laptops, where they can see their schedule this week and the week after. From there they can send a notice for not being able to make it on one of the days or being late. They can also go into their dashboard to request Holidays or study leaves. They can also contact the administrator from there if they want to get in contact. Nurses can also download an Excel or a PDF version of their rota |
| Administrators | Nurse Managers/Nursing Leads are responsible for organising the nurses’ schedules and resolving conflicts between the hospital’s interests and the nurses’ preferences. They have a responsibility to make sure that patient care is provided seamlessly and is not affected by any nursing issues. | Administrators would be interested in decreasing the time they spend on organising the rota and having to deal with the hassle of making sure it complies with the hospital rules and wider NHS laws. Administrators will also not need to individually approve every instance of a nurse requesting a study leave or a bank holiday, unless in exceptional cases e.g. short notice. | They will be able to view the schedule, add and remove nurses, and make changes to the timetable. |
| Hospital Managers | Hospital managers are not involved in offering clinical care but must ensure patient safety and the efficient running of the hospital. They are responsible for recruiting a suitable number of nurses, so the clinical leads have enough personnel to cover all the shifts needed. Hospital managers are also legally obliged to try and save the NHS funds, so they need to ensure that they are not paying any unnecessary or extra wages. | Their rotas will be sure to comply with rules and regulations. The rotas will be sure to optimise for saving the hospital's funds so no shifts would be overstaffed or understaffed, so the hospital needs to employ agency nurses who are more expensive than the in-house nurses. | They would not have direct access to the software but can ask the admin to introduce new constraints on the timetable to be in line with any new guidelines. |
| General Public | The public uses the services of the NHS, it is in their interest for the hospital to run efficiently and be well-staffed. The public also expects the NHS to be efficient with its resources as it uses tax funding from the government. | The public would not directly use the software but would be interested in the benefits it offers to the nurses. Nurses who feel they are well-staffed and not overworked are likely to be better focused on the patient experience and offer better service to their patients. This will reflect positively on the image of the NHS as patients will feel better cared for. | NA |

## Investigating Existing Solutions

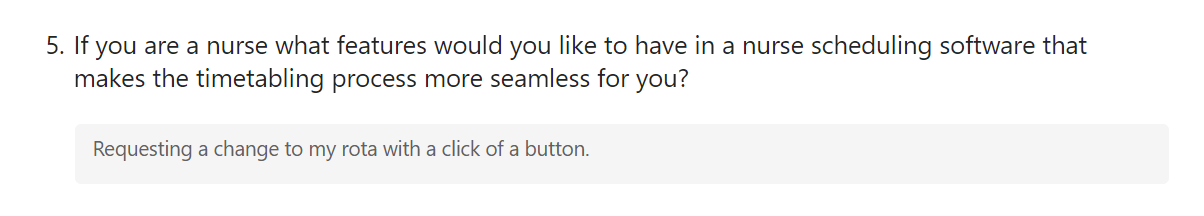
|  |  |  |
| --- | --- | --- |
| Existing Solution | Advantages | Disadvantages |
| Manual Scheduling | * Customisation: Manual scheduling allows for an elevated level of customisation as individual preferences can be accommodated for and understood. * Flexibility: The scheduler can easily make real-time adjustments based on unexpected events, last-minute changes, or emergencies, without relying on complex software. * Personal Understanding: The human touch in scheduling allows for better understanding and consideration of individual needs, which can contribute to improved morale and job satisfaction among nurses. | * Time-Consuming: Creating schedules manually can be time-consuming, especially for large healthcare facilities with many nurses. * Error-Prone: Human errors are more likely to occur in manual scheduling, leading to mistakes in shifts, hours, or days off. These errors can result in dissatisfaction among nurses and potential disruptions in patient care. These errors can also break laws regarding the number of working hours per week and cause legal issues for the hospital. * Limited Optimisation: Manual scheduling may not efficiently optimise staffing levels based on factors such as workload, skills, and preferences. This can lead to suboptimal allocation of resources and potential burnout among nurses. |
| Excel Scheduling | * User-Friendly Interface: Excel provides a familiar spreadsheet interface that is easy to navigate and manipulate, making it accessible to users with varying levels of technical expertise. * Flexibility in Data Representation: Excel allows users to represent scheduling data in tabular format, with rows for shifts, columns for days, and cells for assigning nurses to specific shifts. * Basic Calculation and Formulation: Excel supports basic mathematical operations and formulae, enabling users to perform calculations for shift assignments, total hours worked, and compliance with scheduling rules. * Visualisations and Reporting: Excel offers tools for creating charts, graphs, and pivot tables, allowing users to visualise scheduling trends, identify patterns, and generate reports for analysis. | * Limited Automation: Excel's automation capabilities are limited compared to dedicated scheduling software or programming languages, requiring manual intervention for most scheduling tasks. * Susceptible to Errors: Manual data entry and manipulation increase the risk of errors, such as incorrect shift assignments, overlapping schedules, and inconsistent data formatting. * Difficulty in Handling Complexity: Excel may struggle to handle complex scheduling requirements, such as managing multiple constraints, optimising schedules, and handling large datasets efficiently. |
| External Software | * Efficiency and Time Savings: Automatic schedule builders can quickly generate schedules based on predefined rules, preferences, and constraints, saving considerable time compared to manual scheduling processes. * Optimisation of Resources: These tools often incorporate algorithms to optimise nurse assignments, minimise overtime, balance workloads, and ensure adequate coverage, leading to more efficient resource utilisation. * Adherence to Regulations and Policies: Automatic schedule builders can enforce legal regulations, labour policies, and union agreements, ensuring compliance with labour laws and organisational policies. * Flexibility and Adaptability: Many schedule builders offer flexibility to accommodate changing staffing requirements, unexpected absences, and last-minute scheduling adjustments, ensuring continuity of care and operational efficiency. | * Initial Setup and Configuration: Implementing an automatic schedule builder requires initial setup, configuration, and customisation to reflect the hospital's specific requirements, policies, and constraints. * Complexity and Learning Curve: Using these tools effectively may require training and familiarity with scheduling algorithms, software features, and optimisation techniques, posing a learning curve for users. * Limited Customisation and Control: Some automatic schedule builders may offer limited customisation options, restricting users' ability to fine-tune schedules or accommodate unique scheduling preferences and requirements. * Cost Considerations: Implementing and maintaining automatic schedule builders may incur costs associated with software licensing, support services, and ongoing updates, which hospitals need to evaluate against potential benefits. |

## Questionnaire Results

### Nurse 1:

A screenshot of a computer

Description automatically generated

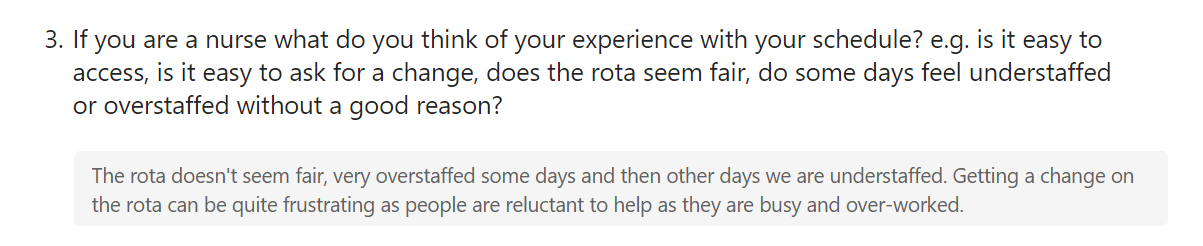


### Nurse 2:

A screenshot of a computer

Description automatically generated

Nurse 3:



A screenshot of a phone

Description automatically generated

### Admin 1:

A close up of a text

Description automatically generated

A screenshot of a computer

Description automatically generated

## Questionnaire Analysis

### Trends in Nurses

* Ease of Access:
  + Most nurses find their schedules easy to access as they are available online, indicating a preference for digital accessibility.
* Requesting Changes:
  + Nurses express difficulty in requesting changes to their schedules, citing the need to do so far in advance (3 months prior). This requirement poses challenges during emergencies or unexpected circumstances.
* Fairness of Rota:
  + There is a consensus among nurses that the rota is not always fair, as some individuals do not always receive the schedules they desire.
* Understaffing/Overstaffing:
  + Nurses note instances of both understaffing and overstaffing on certain days, indicating a lack of optimal staffing balance in the schedules.

### Trends in Admin

* Staff Profiles and Time-off Requests:
  + Administrators prioritise features such as staff profiles for time-off requests and availability, suggesting a need for centralised management of staff scheduling and leave.
* Ease of Schedule Management:
  + There is a preference for features that facilitate easy schedule duplication and viewing of staff holiday entitlement, streamlining administrative tasks, and improving efficiency.

## Features of the proposed solution

|  |  |  |
| --- | --- | --- |
| Feature | Essential or Desirable | Description |
| Automated Optimised Schedule Generation | Essential | Automatically generates schedules based on predefined rules, constraints, and preferences. This will lower manual effort and allow for the use of an optimisation algorithm e.g. Genetic Algorithm |
| Constraint Handling | Essential | Manage various scheduling constraints such as nurse availability, legal regulations, and staffing policies to ensure compliance. The software needs to be able to adapt the schedule based on any changes to the constraints given by the administrator. |
| Database Integration | Essential | The software will need to Integrate with a database to store nurse profiles, scheduling data, preferences, historical records, and legal constraints. |
| GUI for User Interaction | Essential | Easy to use and intuitive graphical interface for nurses and administrators to interact with the scheduling system, view schedules, submit preferences, and adjust the schedules. |
| Different Interfaces for Nurses or Administrators | Essential | Allows for a level of abstraction, e.g. nurses do not need to access the extra functions available to the administrator. |
| User Authentication (With a password or a Biometric method) | Desirable | User authentication mechanisms verify user identity and access rights, ensuring that only authorised users can view, modify, or generate schedules. |
| Preference Management | Desirable | Allows nurses to specify preferences for shifts, days off, working hours, and other scheduling parameters to accommodate individual preferences and improve job satisfaction. |
| Real-Time Updates and Notifications | Desirable | Be able to send real-time updates and notifications to nurses and administrators about schedule changes, shift assignments, pending requests, and other relevant information. |
| Reporting and Analytics | Desirable | Generate reports and provide analytical insights into scheduling trends, resource utilisation, overtime costs, compliance metrics, and other key performance indicators for informed decision-making and cost reduction. |
| Integration with External Systems | Desirable | integrate with external systems such as payroll software and HR management systems to streamline data exchange and ensure data consistency. |
| Customisation and Configuration | Desirable | Offers customisation options in terms of the appearance of the GUI e.g. dark mode. Can also offer configurable settings so each hospital can customise the solution to their individual needs. |
| Encryption | Desirable | Encrypt data about the nurses and which shifts they are working on which days. This protects the privacy of hospital staff; in case the system was compromised in any way. |

## Limitations Of Proposed Solution

|  |  |  |
| --- | --- | --- |
| Limitation | Reasons for non-implementation | Impact |
| My genetic algorithm will assume that all nurses have the same skill level and can work the same shifts interchangeably. | Trying to encode the different skills of the nurses into the algorithm would be complex and introduce extra complications in the mutating and crossing-over steps. The mathematical complexity of this is above the level of my studies and would be very time-consuming to try to implement. | The software might not be very realistic to use in a large hospital with many different wards. However, the algorithm can be still used in smaller single-speciality hospitals or can be used on a single ward within a hospital. |
| The algorithm evaluates that all types of shifts are the same. Even though, many nurses might see night shifts as more challenging and stressful because of their anti-social nature. | Fully addressing nurse preferences and recognising that night shifts are less liked than morning shifts, can be hard to code in a genetic algorithm because of the mathematical complexity of the design; like try coding distinct levels of nurses. | This should not impact the quality of the schedule as labour laws would not allow a nurse to work more than 3 nights in a row, and not more than a specific number in a month. All of this would be considered by the algorithm; however, the algorithm might leave the nurse with 10 days of night shifts which can be stressful and demoralising. |
| Integration with Existing IT Systems in the hospital | Integrating the scheduling system with existing hospital systems (e.g., payroll, HR management) may be complex and require coordination with IT departments, this is very impractical to try and implement now as it needs to be specific to the hospital. | This will not impact any of my success criteria. |
| Data Privacy and Security | Storing sensitive information such as nurse schedules and personal preferences may raise concerns about data privacy and security, requiring complex encryption and security measures to protect sensitive data from unauthorised access or breaches. | The software is going to be tested with sample data, so no personal information would be compromised. As the software would not be used in real hospital environments, the data privacy aspect can be developed later and is not essential to this current stage of software development. |
| Constraints must be entered manually by the administrator because the system is not able to directly access NHS standards. | As the software is not approved by NICE, it would not meet the standards to be connected to NHS IT systems so there is no way for the software to access the newest guidelines without an admin inputting them manually. | This will add a small amount of upfront manual work for the administrator, but it will be massively outweighed by the time-saving benefits of using the software rather than scheduling by hand. |
| Modern looking Interface | The focus is on designing a functioning and easy-to-use interface rather than focusing on aesthetics, as this will be time-consuming to develop and delay the development process of other sections of the project. | This would not impact the stakeholders, as the GUI would be designed to be simple and intuitive to use, which is more important than flare especially as the main of the project is to highlight the use of the Genetic Algorithm approach to solve the nurse scheduling Problem. |

## Hardware And Software Requirements

My solution must be light enough to be run on a variety of devices especially as Hospital IT systems might be old or slow. However, my solution would need Python installed on the computer to be able to run. That means that my solution will be able to run on any platform supporting Python.

## Measurable Success Criteria

|  |  |  |
| --- | --- | --- |
| Code | Criterion | Testing Method and proof |
| 1.1 | The schedule generation algorithm accurately interprets and applies predefined rules. | * Use a reward-punishment system to punish schedules that break any rules and evaluate their different scores to get the highest scoring algorithm. * Use a verification function to make sure none of the hard constraints are violated. * Keep a record of the different iterations of the schedules and include screenshots of their evaluation. |
| 1.2 | The schedule generation algorithm considers nurse availability and shift preferences in assignments. | * Use a similar reward-punishment system to make sure as little as possible soft constraints are violated. * Use a verification algorithm to make sure optimising for the soft constraints did not violate any hard constraints and give them priority. |
| 1.3 | The Schedule optimisation algorithm produces feasible and balanced nurse assignments. | * Conduct a second questionnaire to ask nurses if they feel the schedules produced by the algorithm are realistic and comfortable for them. |
| 1.4 | The optimisation algorithm efficiently allocates nurses to shifts, minimising gaps and overtime. | * Design a function to try and calculate the costs from overtime or locum agency costs that result from the schedule designed by the algorithm and compare the cost to an average cost from NHS trusts. |
| 2.1 | The system accurately captures and records nurse shift preferences. | * Distribute preference surveys to nurses and cross-reference their responses with scheduled assignments to validate accuracy. |
| 2.2 | Nurse shift preferences influence schedule assignments as intended. | * Analyse schedule assignments to ensure that nurses' preferred shift types are prioritised and accounted for appropriately. * Keep a count of all the instances they were not prioritised and have a screenshot of the tracker |
| 2.3 | Nurses can update and modify their shift preferences as needed. | * Provide functionality within the system for nurses to input and update their shift preferences and validate its functionality. |
| 2.3.1 | Screen from which a nurse can request a day off by providing a reason. | * Screenshot |
| 2.3.1.1 | If the notice from the nurse is longer than 3 weeks this will be automatically reflected on the schedule | * Screenshots from the schedule before and after the change was requested |
| 2.3.1.2 | if the notice is shorter than 3 weeks it will flag up to the administrator to review it and authorise the action. | * Screenshot from an admin account to verify that they got the message and can approve it |
| 3.1 | Access control restricts user permissions based on their roles and responsibilities. | * Screenshot of the user interface from a nurse's perspective and a screenshot from the administrator’s perspective. |
| 4 | The user interface design is clear, consistent, and usable. | * Conduct a questionnaire with nurses to see if they think the layout follows these principles. |
| 4.1 | Navigation menus and buttons are intuitively organised and labelled. | * Observe a user interact with the system and ask them to conduct tasks to see if they can navigate their way through the system. |
| 4.2 | Error messages are informative and guide users towards resolution. | * Create an error scenario and observe a stakeholder trying to solve it by following the error messages. |
| 4.3 | GUI layout and design are responsive and compatible with a range of screen sizes. | * Screen recordings of me trying to resist the window and still confirming that the main functionality works |
| 5.1 | Database Creation | * Screenshot of the empty database with no error messages |
| 5.2 | Data Population: Populate the database with initial data, including nurse profiles, shift schedules, and organisational information. | * Screenshots of the populated database and compare it with the test data to make sure everything |
| 6.2 | CRUD Operations Support: Implement support for CRUD (Create, Read, Update, Delete) operations to enable data manipulation within the application. | * Screenshots of the database before and after operations on the database and compare them with the test data to make sure the correct data was updated. |
| 6.3 | Query Performance Optimisation | * Keep a record of the time taken to query the database before and after the performance optimisation algorithm |

# Design

## Algorithms

The core algorithm in my project will be a Genetic algorithm to try and find the most optimal schedule. A genetic algorithm aims to use concepts from biological evolution to find the “fittest” option. The algorithm involves 3 steps: the selection process, crossover and mutation, and fitness evaluation. There will be supportive algorithms to help optimise some parts of the main algorithm, these algorithms would do more specific tasks. The potential extra algorithms might contain a greedy initialisation algorithm, local search algorithm, simulated annealing (SA), constraint propagation algorithm and database query optimisation. The genetic algorithm is central to my project because it’s ideal for tackling the complexity of nurse scheduling with its iterative, adaptive approach. The Supplementary algorithms help manage the initial setup and edge cases. The algorithm will consider both hard constraints (e.g., coverage requirements) and soft constraints (e.g., nurse preferences) to produce feasible and fair schedules, the final step of the scheduling process must involve a human administrator to make sure the schedule applies to real-life scenarios.

### Genetic Algorithm

**Module:** GeneticAlgorithm

**Class:** GeneticAlgorithm

The core Genetic Algorithm (GA) will handle the optimisation of nurse schedules. It will work by evolving a population of possible schedules over several generations, selecting the best candidates, and using crossover and mutation to produce new candidates. This algorithm will interact with supporting algorithms like fitness evaluation, selection, crossover, and mutation.

#### Pseudo code:

Class GeneticAlgorithm :

Function initialise\_population(pop\_size):

# Initialise a population of schedules

population = []

for i from 1 to pop\_size:

schedule = generate\_random\_schedule()

population.append(schedule)

return population

Function evaluate\_fitness(population):

# Calculate the fitness of each schedule in the population

for schedule in population:

schedule.fitness = calculate\_fitness(schedule)

return population

Function select\_parents(population, selection\_method):

# Select parents based on their fitness

parents = []

while len(parents) < required\_parents:

parent = selection\_method(population)

parents.append(parent)

return parents

Function crossover(parents):

# Combine parents to produce offspring

offspring = []

for i from 0 to len(parents) step 2:

child1, child2 = crossover\_method(parents[i], parents[i+1])

offspring.append(child1)

offspring.append(child2)

return offspring

Function mutate(offspring):

# Apply random mutations to offspring

for a child in offspring:

mutate\_method(child)

return offspring

Function evolve(pop\_size, generations):

# The main loop of the genetic algorithm

population = initialise\_population(pop\_size)

for generation from 1 to generations:

population = evaluate\_fitness(population)

parents = select\_parents(population, selection\_method)

offspring = crossover(parents)

offspring = mutate(offspring)

population = select\_next\_generation(population, offspring)

return get\_best\_schedule(population)

### The supporting algorithms

#### Greedy Initialisation Algorithm

The Greedy Initialisation Algorithm generates an initial population of schedules by making a series of locally optimal choices. It aims to quickly create a set of feasible schedules that can be refined by the main Genetic Algorithm. This approach is used to provide a good starting point for optimisation. As one of the principles in a genetic algorithm that the quality of the imput greatly affects the quality of the ouput.

##### Pseudocode:

Class GreedyInitialiser:

Function generate\_initial\_population(pop\_size):

# Initialise population with greedy method

population = []

while len(population) < pop\_size:

schedule = generate\_greedy\_schedule()

if is\_feasible(schedule):

population.append(schedule)

return population

Function generate\_greedy\_schedule():

# Create a schedule by making the best choice at each step

schedule = empty\_schedule()

while not schedule\_complete(schedule):

shift = choose\_best\_shift(schedule)

assign\_shift(schedule, shift)

return schedule

Function choose\_best\_shift(schedule):

# Select the best shift to assign based on current state

best\_shift = None

best\_value = -infinity

for shift in available\_shifts:

value = evaluate\_shift(schedule, shift)

if value > best\_value:

best\_value = value

best\_shift = shift

return best\_shift

Function is\_feasible(schedule):

# Check if the schedule meets all constraints

return all\_constraints\_met(schedule)

#### Local Search Algorithm

The Local Search Algorithm iteratively improves a schedule by making small changes and evaluating their impact. It seeks to find a better schedule by exploring local changes and checking if they lead to a better solution. This algorithm is useful for refining schedules generated by other methods. The Local Search Algorithm can be used to refine the schedules produced by the crossover and mutation processes. It helps to further optimise these schedules by making small, local improvements and finding better solutions within the neighbourhood of each schedule.

##### Psudocode:

Class LocalSearcher:

Function perform\_local\_search(initial\_schedule):

# Iteratively improve the schedule

current\_schedule = initial\_schedule

while not at\_optimal(current\_schedule):

neighbour = find\_best\_neighbour(current\_schedule)

if evaluate\_fitness(neighbour) > evaluate\_fitness(current\_schedule):

current\_schedule = neighbour

return current\_schedule

Function find\_best\_neighbour(schedule):

# Generate and evaluate neighbours

best\_neighbour = None

best\_value = -infinity

for neighbour in generate\_neighbours(schedule):

value = evaluate\_fitness(neighbour)

if value > best\_value:

best\_value = value

best\_neighbour = neighbour

return best\_neighbour

Function generate\_neighbours(schedule):

# Create neighbouring schedules by making small changes

neighbours = []

for change in possible\_changes:

neighbour = apply\_change(schedule, change)

neighbours.append(neighbour)

return neighbours

#### Simulated Annealing (SA) Algorithm

The Simulated Annealing Algorithm is a probabilistic technique that searches for a good solution by exploring the schedule space. It starts with a high "temperature" to allow exploration of various solutions and gradually cools down to refine the search. This method helps avoid local minima and aims to find a near-optimal solution. Simulated Annealing can be used as an alternative or complementary method to Local Search for exploring the solution space. It helps in avoiding local minima by probabilistically accepting worse solutions at higher temperatures, thus potentially leading to better overall solutions.

##### Pseudocode:

Class SimulatedAnnealer:

Function perform\_simulated\_annealing(initial\_schedule, temperature, cooling\_rate):

current\_schedule = initial\_schedule

best\_schedule = current\_schedule

while temperature > minimum\_temperature:

neighbour = generate\_random\_neighbour(current\_schedule)

delta = evaluate\_fitness(neighbour) - evaluate\_fitness(current\_schedule)

if delta > 0 or random() < exp(delta / temperature):

current\_schedule = neighbour

if evaluate\_fitness(current\_schedule) > evaluate\_fitness(best\_schedule):

best\_schedule = current\_schedule

temperature \*= cooling\_rate

return best\_schedule

Function generate\_random\_neighbour(schedule):

# Create a random neighbour by making a random change

change = random\_change()

return apply\_change(schedule, change)

#### Constraint Propagation Algorithm

The Constraint Propagation Algorithm ensures that constraints are satisfied by propagating the effects of constraint violations through the schedule. It adjusts the schedule dynamically as constraints are enforced, helping to reduce the search space by eliminating infeasible solutions early on. Propagation is used During and After Initialisation and Local Refinements to enforce constraints and ensure that generated schedules adhere to all the problem requirements. This helps in reducing infeasible schedules early on and can improve the quality of schedules generated by the genetic algorithm.

##### Pseudocode:

Class ConstraintPropagator:

Function propagate\_constraints(schedule):

# Enforce constraints and propagate changes

while constraints\_remaining(schedule):

for constraint in constraints:

if is\_violation(constraint, schedule):

schedule = resolve\_violation(constraint, schedule)

return schedule

Function resolve\_violation(constraint, schedule):

# Adjust the schedule to resolve a constraint violation

if constraint.type == 'shift':

adjust\_shifts(schedule, constraint)

elif constraint.type == 'day\_off':

adjust\_days\_off(schedule, constraint)

# Other constraints can be handled similarly

return schedule

## Usability Features

|  |  |  |
| --- | --- | --- |
| Feature | Description | Relevance |
| Clear Simple Layout | An easy-to-navigate interface with a clear, logical layout. Consistent use of design elements and labels. | Ensures that users can quickly understand and use the system without confusion, enhancing overall efficiency and user satisfaction.` |
| Role-Based Access | Different functionalities and interfaces based on user roles (e.g., nurses and administrators). | Customises the experience to the specific needs and permissions of each user type, improving security and usability. |
| Automated Scheduling | Automatic generation of schedules based on predefined rules and algorithms. | Reduces manual input, minimises errors, and ensures schedules are optimised according to constraints and preferences. |
| Export Capabilities | Ability to export schedules to formats like Excel. | Facilitates integration with other systems and allows for flexible data management and reporting. |
| Data Security | Mechanisms for protecting sensitive data through access controls. | Safeguards user information and maintains privacy and compliance with data protection regulations. |

## Main Screens[[2]](#footnote-3)

A screenshot of a login screen

Description automatically generated

This is the login screen it’s designed to be as clear and simple as possible to allow as many nurses as possible to understand it and minimise any distractions. The credentials will enable the program to employ access control giving the nurses different levels of access compared to the administrators.

A screenshot of a computer

Description automatically generated

This will be the only part of the program accessible by the nurse class users as it will allow them to see their schedule and update their preferences as needed. The Schedule window has three main functionalities: Viewing Schedules, adding and removing preferences and planning schedules(admins only). The schedule is displayed in a weekly format with the user able to see one-week forward and one week backward of needed. In our program, the use will also be able to output the schedule into an excel sheet or pdf.

A screenshot of a computer

Description automatically generated

The preferences function allows a nurse to enter what days they are available and what times on these days of everything is unchecked as seen in the image then the program assumes that the user doesn’t have any special preferences.

A screenshot of a computer

Description automatically generated

The user window is only accessible by the administrator and allows them three main functionalities: Add user, Modify User, and Remove User. Adding the user requires all the information to be entered as per legal requirements otherwise the new nurse wouldn’t be saved if some of the data is missing.

A screenshot of a computer

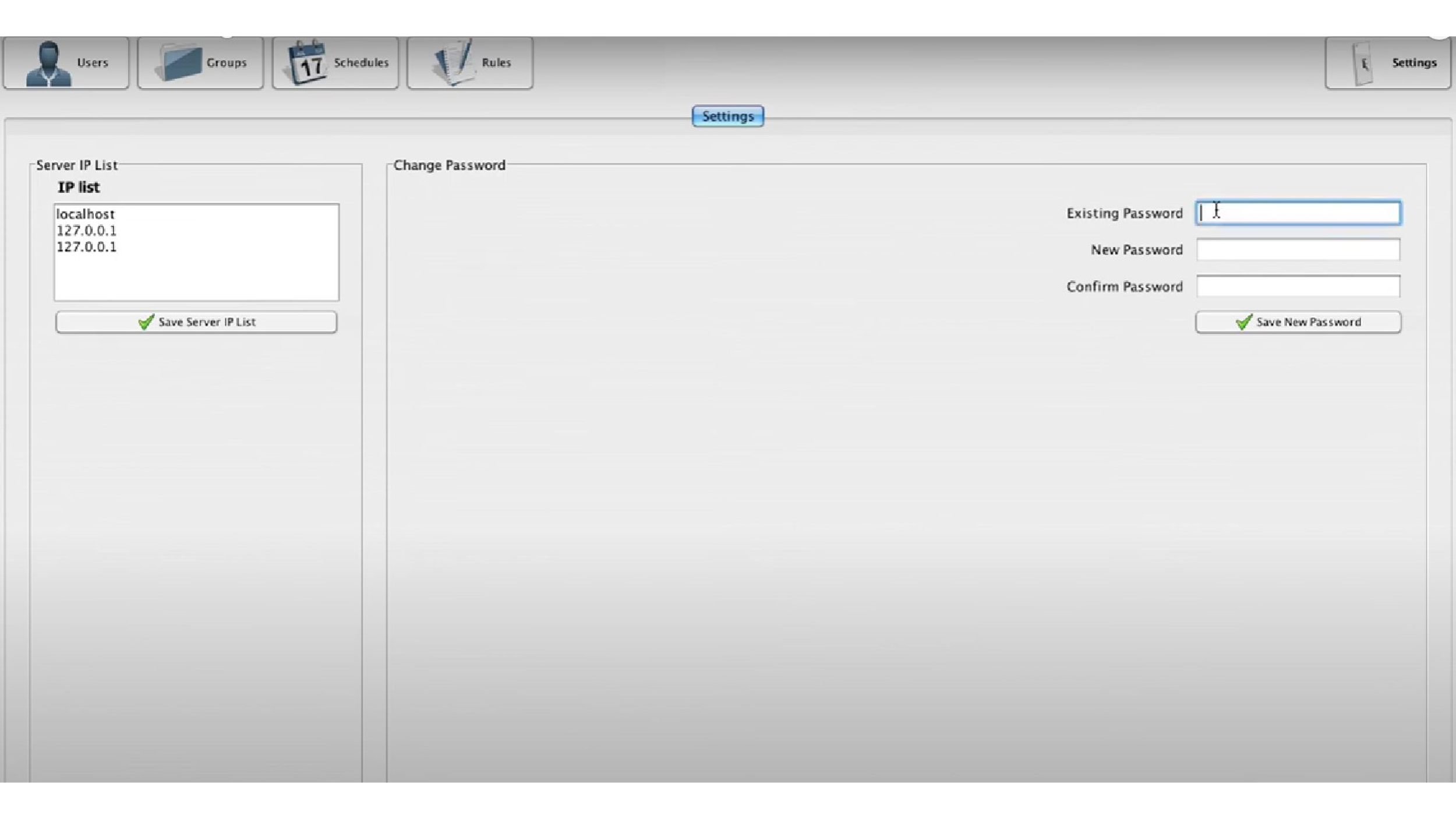
Description automatically generated

The modify users function allows the administrator to change details about nurse already on the database. This allows the administrator to quickly make sure all the information about the nurses are accurate and up to date. The window is designed to be as simple and intuitive as possible to limit the training time required and assist healthcare staff rather than slow them down.

A screenshot of a computer

Description automatically generated

Removing a user is simpler than adding or modifying a user as it only requires the admin to search the nurse id if known and the record relating to that id will automatically be deleted from the database.



This page is just to manage the access control system and can’t currently handle if the user forgets their password. The part about the IP address will not be shown to the user as it adds a layer of unneeded confusion about how to use the system.

## Key Data Structures to be Used and Design of the Database:

### Data Structures

The main data structure to be used is dictionaries, the program will rely on 4 main dictionaries to store: nurse profiles, schedules, constraints and requests and preferences. The nurse profile dictionary Stores individual nurse details with Each nurse identified by a unique ID. The schedule dictionary stores the generated schedules, mapping shifts to assigned nurses. The constraints dictionary stores constraints related to shift assignments, to ensure compliance with staffing rules and regulations and prevent overworking or underworking of staff. The preferences dictionary stores individual requests and preferences from nurses.

|  |  |
| --- | --- |
| Dictionary | Basic Structure |
| Nurse Profiles | nurse\_profiles = {nurse\_id: {'name': str, 'skills': list, 'availability': dict, 'preferences': dict, 'requests': list}} |
| Schedules | schedules = {shift\_id: {'date': str, 'shift\_type': str, 'assigned\_nurses': list}} |
| Shift Constraints | shift\_constraints = {'min\_shifts\_per\_nurse': int, 'max\_shifts\_per\_nurse': int, 'max\_consecutive\_shifts': int, 'min\_days\_off': int} |
| Requests and Preferences | requests\_and\_preferences = {nurse\_id: {'requests': list, 'preferences': dict}} |

### Database

A close-up of several white squares

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field Name | Data Type | FK or PK |
| NURSES | nurse\_id | ID | PK |
|  | first\_name | VARCHAR |  |
|  | last\_name | VARCHAR |  |
|  | skills | VARCHAR[] |  |
|  | availability | JSON |  |
| SCHEDULES | schedule\_id | ID | PK |
|  | nurse\_id | ID | FK |
|  | shift\_date | DATE |  |
|  | shift\_time | TIME |  |
|  | shift\_type | VARCHAR |  |
| PREFERENCES | preference\_id | ID | PK |
|  | nurse\_id | ID | FK |
|  | preferred\_shifts | VARCHAR[] |  |
|  | unavailable\_days | DATE[] |  |
|  | special\_requests | TEXT |  |
| ADMINISTRATORS | admin\_id | ID | PK |
|  | first\_name | VARCHAR |  |
|  | last\_name | VARCHAR |  |
|  | role | VARCHAR |  |
|  | permissions | VARCHAR[] |  |

## Test data

### The main objectives of the system and how the testing process addresses them?

|  |  |  |
| --- | --- | --- |
| Success Criterion | Test Description | NHS Value Addressed |
| The system should ensure fair and balanced shift distribution. | Test with varied nurse preferences and availabilities to check for fair distribution of shifts. | Staff Well-being, Patient Safety |
| The system must accommodate emergency shift swaps efficiently. | Test the ability of nurses to request and receive quick approvals for shift swaps. | Staff Well-being |
| Shifts must align with NHS guidelines for rest periods between shifts. | Test if the system schedules nurses with appropriate rest periods in between shifts. | Patient Safety, Staff Well-being |
| The system should prevent understaffing or overstaffing on any given shift. | Run tests with different staffing levels to ensure optimal staffing based on demand. | Patient Safety, operational efficiency |
| Nurses must have easy access to their schedules. | Test the user interface for accessibility and ease of schedule viewing. | Staff Well-being |
| Administrators should have an efficient interface for managing schedules. | Test the admin interface for adding, removing, and modifying schedules quickly. | Staff Well-being |
| Ensure compliance with legal working hours and overtime rules. | Validate the scheduling algorithm against NHS working hours policies to ensure compliance. | Patient Safety, Staff Well-being |

### The main tests to test the robustness of the system

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test Data Description | Justification | Expected Outcome |
| 1 | Single Nurse Schedule: A single nurse with no prior assignments | Validates the ability of the system to create a new schedule for a nurse without any conflicts. This is the most basic scenario. | The system should successfully create a schedule for the nurse without any errors or conflicts. |
| 2 | Multiple Nurses with Different Availability: Varying shift availability | Tests how the system handles multiple nurses with different shift preferences and availability, ensuring that all nurses get assigned appropriately. | The system should correctly assign shifts according to each nurse’s availability and preferences. |
| 3 | Overlapping Schedules: Two nurses with overlapping shift requests | Ensures that the system can resolve conflicts when multiple nurses request the same shift, particularly in overstaffed or understaffed conditions. | The system should detect the conflict and resolve it according to the fairness criteria (e.g., skill level, rotation). |
| 4 | Edge Case - Maximum Capacity: Schedule that pushes the system's capacity (e.g., maximum number of nurses or shifts) | Validates the robustness of the system when operating at full capacity, testing performance and stability. | The system should continue to function correctly without crashing, providing appropriate error messages if necessary. |
| 5 | Invalid Data Input: Non-numeric nurse ID, empty fields, and incorrect data types | Tests the system’s input validation to ensure that it handles erroneous or malicious inputs gracefully without crashing or generating incorrect outputs. | The system should reject invalid inputs with clear error messages and prevent the creation of faulty schedules. |
| 6 | Historical Data Query: Retrieve schedules from previous months/years | Tests the efficiency and accuracy of the system’s ability to query and retrieve historical scheduling data, ensuring data integrity over time. | The system should return accurate historical data quickly, verifying data integrity over extended periods. |
| 7 | Holiday Requests: Nurses requesting holidays during high-demand periods | Validates how the system manages holiday requests, ensuring fairness and maintaining sufficient staffing levels during critical periods. | The system should correctly log holiday requests and manage staffing levels without compromising service quality. |
| 8 | Preference Overrides: Test if the system correctly handles overriding preferences when critical staffing is required | Ensures that the system can override personal preferences in emergency or critical situations while minimising dissatisfaction. | The system should handle preference overrides smoothly and fairly, with appropriate notifications to affected nurses. |
| 9 | Simultaneous Shift Swaps: Multiple nurses requesting shift swaps simultaneously | Tests the system’s ability to handle and resolve multiple shift swap requests at once, ensuring no double bookings or gaps. | The system should manage all swap requests effectively, ensuring a balanced and complete schedule post-swaps. |

### Exemplar nurse data to be used

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nurse ID | Name | Age | Availability | Preferred Shifts | Assigned Shifts |
| 1 | Alice Johnson | 34 | Mon-Fri (Morning), Sat (Evening) | Morning Shifts |  |
| 2 | Bob Smith | 45 | Mon-Sun (Flexible) | Night Shifts |  |
| 3 | Claire Williams | 29 | Mon, Wed, Fri (Morning), Sun (Night) | Morning Shifts |  |
| 4 | David Brown | 52 | Tue, Thu, Sat (Evening), Sun (Morning) | Evening Shifts |  |
| 5 | Emily Davis | 27 | Mon-Wed (Afternoon), Fri (Morning) | Afternoon Shifts |  |
| 6 | Frank Miller | 38 | Mon, Tue, Thu (Night), Sat (Morning) | Night Shifts |  |
| 7 | Grace Taylor | 31 | Mon-Fri (Morning), Sat-Sun (Flexible) | Morning Shifts |  |
| 8 | Henry Wilson | 43 | Mon-Fri (Night), Sat (Afternoon) | Night Shifts |  |
| 9 | Isabella Moore | 26 | Mon-Wed (Morning), Thu-Sun (Flexible) | Morning Shifts |  |
| 10 | Jack Thompson | 49 | Mon-Fri (Evening), Sat (Morning) | Evening Shifts |  |
| 11 | Kelly Anderson | 35 | Tue, Thu, Sat (Afternoon), Sun (Morning) | Afternoon Shifts |  |
| 12 | Luke Harris | 41 | Mon-Fri (Morning), Sat-Sun (Flexible) | Morning Shifts |  |
| 13 | Maria Clark | 33 | Mon-Wed (Afternoon), Fri-Sun (Morning) | Morning Shifts |  |
| 14 | Nathan Lewis | 30 | Mon, Wed, Fri (Morning), Sun (Night) | Morning Shifts |  |
| 15 | Olivia Scott | 28 | Mon-Fri (Evening), Sat (Afternoon) | Evening Shifts |  |
| 16 | Peter Young | 54 | Mon-Wed (Night), Thu-Sun (Morning) | Night Shifts |  |
| 17 | Rachel King | 39 | Mon-Fri (Morning), Sat-Sun (Flexible) | Morning Shifts |  |
| 18 | Simon Wright | 42 | Mon-Wed (Afternoon), Thu-Sun (Morning) | Afternoon Shifts |  |
| 19 | Theresa Walker | 37 | Mon-Fri (Morning), Sat-Sun (Night) | Night Shifts |  |
| 20 | Victor Hall | 36 | Mon-Wed (Night), Thu-Sun (Afternoon) | Night Shifts |  |
| 21 | Wendy Allen | 32 | Mon-Fri (Evening), Sat (Morning) | Evening Shifts |  |
| 22 | Xavier Wright | 40 | Mon-Fri (Morning), Sat (Afternoon) | Morning Shifts |  |
| 23 | Yvonne Evans | 29 | Mon-Wed (Afternoon), Fri-Sun (Morning) | Afternoon Shifts |  |
| 24 | Zachary Adams | 44 | Mon-Fri (Night), Sat (Morning) | Night Shifts |  |
| 25 | Abigail Phillips | 25 | Mon-Wed (Morning), Thu-Sun (Flexible) | Morning Shifts |  |

## Top-down Decomposition.

## 

1. Hakim, L., Bakhtiar, T. and Jaharuddin (2017) ‘The nurse scheduling problem: A goal programming and nonlinear optimisation approaches’, IOP Conference Series: Materials Science and Engineering, 166, p. 012024. doi:10.1088/1757-899x/166/1/012024. [↑](#footnote-ref-2)
2. The screenshots came from a project designed by California State University students and published on a YouTube video. I asked for permission from the owner of the project to use the screenshots, the workings behind the system are very different so I only got access to the screenshots from the video but no other help in anyway. [↑](#footnote-ref-3)