Documentation and Report team

1. Roles/Responsibilities

Name	Team to Track	Regular roles & responsibilities	Roles after receiving the final reports from each team
Pranav Patel (SV)	Documentation and Reporting User Research Sorting Algo Graph Theory DP	Doc Team Lead Plan workflows, ensure team activity and understanding of roles, and communicate with other team leads for updates.	Ensure all tasks are progressing as planned by delegating subtasks to team members and taking on any tasks that are behind schedule or unassigned. Provide final edits to the draft report.
Muqing Cao (SV)	Sorting Algorithms	Collect info Catch up Keep track Summarize similar papers for **Related Work**	**Project Design** & **Methods** of our final report
Guangmei Xiang (SV)	Graph Theory	Collect info Catch up Keep track Summarize similar papers for **Related Work**	**Discussion**, **Data Collection** & **References**
Himalaya Dua (SV)	Graph Theory/ User Research	Summarize similar papers for **Related Work**	**Related Work**
Qin He (SV)	DP	Collect info Catch up Keep track Summarize similar papers for **Related Work**	**Result** & **Conclusion**
Ning Li (??)	User Research	Keep in touch with team and extract insights from their report	**Abstract** & **Introduction**

Each team member was responsible for maintaining communication with their assigned teams, monitoring their progress, and compiling a comprehensive report. Upon receipt of all team reports, we integrated the pertinent information into the final draft, documenting the collective efforts. Subtopics were divided and allocated as outlined above, with each team member contributing to the successful achievement of the expected outcomes.

2. Expectations

- Keep track of progress
    - 3 implementations
        - sorting,
        - dynamic programming (DP), and
        - graph theory
    - Assign one person to each team to track
        - Contact person (names will be shared by Srija)
        - Collect info
        - Catch up
        - Keep track
        - Decide a meeting schedule (daily, bi-weekly)
    - Remaining does documentation and research
- Methods used
- write the research paper
- document algorithms
- manage presentation materials
- Changes
- Documentation
    - Make a layout of Paper
        - What worked
        - What didn’t
- Research
    - Similar papers
    - Extract related works
        - Style
        - Language
        - Layout

3. Report

Abstract

The healthcare industry faces significant challenges in optimizing scheduling systems to balance limited resources, high patient demand, and competing priorities. This paper presents a collaborative project that integrates sorting algorithms, dynamic programming (DP), and graph theory to design an efficient group-based healthcare scheduling algorithm. Conducted as part of an algorithm course, this project engages students in cross-functional teams to address real-world scheduling problems, such as reducing patient wait times, optimizing resource allocation, and enhancing hospital efficiency. Using surveys, user testing, and performance metrics, the project evaluates educational outcomes, including improved teamwork, algorithmic proficiency, and student satisfaction. Results demonstrate the transformative potential of algorithmic solutions in healthcare and highlight the value of interdisciplinary, industry-aligned educational projects for fostering technical and collaborative skills. Future directions include scaling the system for larger datasets and integrating predictive analytics to further refine healthcare scheduling processes.

Introduction

Efficient healthcare scheduling is crucial for managing limited resources, reducing patient wait times, and improving overall patient satisfaction. Hospitals often face challenges such as high patient volumes, competing priorities between emergency and routine cases, and constraints on available resources like doctors, treatment rooms, and medical equipment. Addressing these challenges requires innovative algorithmic solutions that can optimize resource allocation and streamline patient flow.

This project, conducted within an algorithm course, aims to provide students with a platform to apply theoretical concepts to a practical and impactful problem. By working in interdisciplinary teams, students design and implement algorithms that tackle various aspects of healthcare scheduling. The project incorporates sorting algorithms for patient prioritization, dynamic programming for resource allocation, and graph theory for patient flow optimization. The collaborative nature of the project not only enhances students' algorithmic knowledge but also fosters essential teamwork and problem-solving skills.

In addition to technical development, the project emphasizes educational research through surveys and user testing. These tools evaluate the impact of the project on students’ understanding of algorithms, teamwork dynamics, and satisfaction with the learning process. The findings contribute to the broader goal of improving educational practices in algorithm courses by integrating real-world, industry-relevant challenges.

This paper explores the design, implementation, and outcomes of the project, highlighting its dual contributions to healthcare optimization and algorithmic education. By addressing both technical and educational dimensions, the project demonstrates the potential of collaborative learning approaches to prepare students for future roles in technology and data-driven decision-making.

Related Work

The optimization of healthcare scheduling systems represents a complex challenge requiring sophisticated algorithmic approaches. Three primary methodologies have emerged as particularly promising: sorting-based algorithms, dynamic programming solutions, and graph theory applications.

1. **National Early Warning Score (NEWS):** This track-and-trigger system aggregates patient vital signs into a score ranging from 0 to 20, measured at least twice a day (Smith et al., 2019). The score determines the escalation protocol, influencing monitoring intervals and preventive actions for patient deterioration.
2. **Pareto Optimization in Patient Sorting:** The Pareto front represents trade-offs between conflicting objectives in multi-objective optimization. For example:
    - Objective 1: Minimize the average patient waiting time.
    - Objective 2: Minimize facility completion time (time the last patient leaves). Solutions on the Pareto front ensure no single objective can improve without worsening the other (Saremi et al., 2015). Using this concept, decision-makers can select optimal solutions that balance waiting times and resource efficiency.
3. **Mixed-Integer Linear Programming (MILP) for Healthcare Management:** Healthcare delivery systems are under increasing strain due to rising patient expectations, aging populations, and limited resources. Yinusa and Faezipour (2023) present a MILP model integrating staffing, patient scheduling, and resource allocation. This approach demonstrates the capacity to minimize costs, reduce patient wait times, and ensure equitable distribution of resources through simulated data and advanced algorithms.

Project Design

Overall Scope:

The primary objective of this project is to develop an advanced healthcare scheduling and resource allocation system that optimizes operational efficiency while enhancing patient outcomes. The project integrates sorting algorithms and dynamic programming techniques to manage in-patient and out-patient scheduling effectively.

Key Objectives:

The project focuses on achieving several critical goals. First, it aims to implement an efficient patient prioritization system using a three-tiered sorting algorithm that accounts for medical urgency, scheduled times, and priority scores. Second, dynamic programming techniques are leveraged to optimize room utilization, reduce wait times, and balance resource allocation effectively over multiple days. Scalability is another key objective, with modular and flexible solutions designed to cater to both small-scale and large-scale healthcare settings. Lastly, the project seeks to deliver actionable insights by providing detailed performance metrics, such as wait times and room utilization, to enhance decision-making.

Team Roles:

Sorting Algorithm Team: Develop and test the sorting algorithm based on input data (patient_id, date, condition, etc.). Optimize for accuracy, fairness, and scalability, and document multi-level sorting logic.

Dynamic Programming Team: Implement dynamic programming for room allocation and conflict resolution. Optimize scheduling and prepare documentation with performance metrics.

Graph-Based Optimization Team: Model scheduling as a directed graph and develop algorithms to optimize patient prioritization. Design resource allocation strategies and document results.

User Research Team: Conduct surveys, interviews, and observations to gather feedback. Evaluate impact through metrics like patient satisfaction, wait times, and resource utilization. Provide feedback to guide adjustments in the algorithms.

Documentation Team: Finalize technical documentation with implementation teams. Compile research findings into a paper, highlighting algorithm performance, efficiency, and resource optimization.

Project Alignment with Course Content and Industry Practices:

The project's use of sorting algorithms (e.g., sorting by date, condition, and priority) aligns directly with the course's focus on sorting techniques like quicksort and heapsort. Efficient sorting ensures quick decision-making, a key consideration in real-world healthcare systems for prioritizing patients.

Dynamic programming (DP) is used to optimize room allocation and resolve scheduling conflicts. This aligns with the course's focus on DP, as the project employs bottom-up DP for daily scheduling and recursive DP for long-term planning, directly applicable in industries like healthcare for resource management.

The use of graph-based optimization, where the hospital system is modeled as a directed graph, aligns with the course's coverage of graph algorithms. This modeling allows for the optimization of patient scheduling and resource allocation, a common practice in industries dealing with complex systems, such as transportation and healthcare.

Methods

Sorting Algorithm:

The sorting algorithm was designed to handle both in-patient and out-patient scheduling requirements, which differ significantly. While in-patient scheduling focuses on long-term resource planning, out-patient scheduling addresses shorter consultations with quicker resource turnovers. To accommodate these differences, the algorithm dynamically adapts to prioritize patients effectively.

The algorithm begins by standardizing patient data, including fields such as scheduled date and time, appointment duration, required resources, priority score, condition type, and target wait time. Patients are initially sorted by their scheduled date, followed by their scheduled time for chronological prioritization. Within the same date and time group, patients are further organized by condition type, with emergencies taking precedence over urgent or routine cases. For cases with identical condition types, a priority score is used to finalize the order, ensuring that higher urgency patients are handled first. This three-tiered sorting strategy ensures that the output is a well-ordered list of patients, optimized for resource allocation and operational efficiency.

Dynamic Programming:

For smaller-scale scheduling problems, a time-slot-based dynamic programming approach was employed. Patient schedules were divided into manageable datasets, and a bottom-up dynamic programming strategy was used to allocate rooms efficiently. This method iteratively assigned rooms to patients while ensuring a minimum interval of five minutes between appointments to avoid conflicts. When scheduling conflicts arose, priority was given to patients with higher urgency scores. The team closely monitored performance metrics such as average wait times and the number of violations against set thresholds to refine the algorithm.

For larger-scale, multi-day scheduling, the team developed a modular dynamic programming model. Patient data was grouped and processed to handle complex multi-day scheduling requirements. An initial room allocation was calculated using a greedy algorithm, which provided baseline metrics such as minimum required rooms and average wait times. This was followed by a recursive dynamic programming optimization process, which adjusted room allocations to minimize total costs. The cost function considered factors like room availability, patient priority, and target wait times. A central coordination system integrated data preparation, allocation, and optimization, producing detailed reports and statistical analyses to support decision-making. While the single-class approach was efficient for smaller problems, the modular design provided scalability and flexibility for larger datasets, albeit at the cost of increased computational complexity.

Graph-Based Optimization:

The hospital system was modeled as a directed graph, with nodes representing patients and resources, such as rooms and doctors, and edges indicating assignments. The graph's edge weights were determined by a combination of patient priority scores and doctor efficiencies, ensuring that resource utilization was optimized.

Doctor efficiency was dynamically calculated based on several factors, including base efficiency, overtime penalties, rest penalties, and workload degradation. This approach allowed the team to account for the varying capabilities and constraints of different types of doctors, such as emergency specialists, general practitioners, and consultants. By incorporating these dynamic factors, the team ensured that the scheduling model balanced workload distribution while prioritizing high-urgency cases.

Room assignments were optimized to ensure conflict-free scheduling. The algorithm systematically identified the next available time slot for each patient by iterating through the schedule and accounting for cleaning times and required intervals between appointments. This method minimized idle time for resources while maintaining strict adherence to scheduling constraints.

A pathfinding algorithm was employed to optimize patient flow within the scheduling system. The algorithm maintained a set of patients pending assignment, with scores representing the cumulative cost from the start node and heuristic estimates guiding the optimization process. The final assignments were determined by balancing patient wait times and resource efficiency, with adjustments made dynamically to handle changes in patient priorities or resource availability.

The overall optimization process involved aligning patient scheduling with resource utilization. Patients were prioritized by their urgency and target wait times, while doctors were ranked based on historical workload data and dynamically adjusted efficiency scores. This integrated approach ensured that wait times were minimized and resources were utilized effectively, resulting in a robust and adaptable scheduling system.

Data Collection

The data collection process for the healthcare scheduling optimization project involved two primary components: synthetic dataset generation for algorithm testing and user-based evaluation for system performance and usability. The combination of quantitative metrics and qualitative feedback provided a comprehensive understanding of system efficiency, accuracy, and user satisfaction.

1. Synthetic Dataset

The project generated a synthetic dataset tailored for in-patient scheduling, based on a simplified extraction of NHS sample data. This dataset included healthcare referral and care contact information from various UK health organizations for August 2024. This monthly report, part of the Community Service Data Set (CSDS) available through NHS England Digital, provides a basis for testing algorithms, hypotheses, and methodologies. The data schema integrates patient, doctor, and resource attributes through conditional mapping, ensuring uniform attribute generation across defined ranges. Key attributes include unique patient identifiers, appointment schedules, time slots, duration, required resources, and priority scores, with constraints designed to simulate real-world healthcare scenarios. Priority scores and wait time targets reflect urgency levels, categorized as "Emergency," "Urgent," or "Routine," and are used to minimize scheduling delays while aligning with service level agreements. This dataset enables detailed analysis of scheduling efficiency, patient wait times, and resource utilization in healthcare environments.

Scheme Summary

Attribute	Description	Value/Constraints
patient_id	Unique identifier each patient	Format: P000XXX
scheduled_date	Date of appointment	Format: ‘YYYY-MM-DD’ Range: August 2024
scheduled_time	Time Slot	Format: ‘HH:MM’ Range: 08:00 to 20:00
duration_minutes	Length of appointment based on condition_type	Integer in minutes
required_resources	Required room and staff type based on condition_type	Room, Staff Type
priority_score	Base priority score assessed during the referral process before the appointment is scheduled	Range: Higher score indicates more urgent case:

Attribute	Description	Value/Constraints
		Emergency: 80 ± 10 Urgent: 60 ± 10 Routine: 40 ± 10
Wait_time_target	The maximum acceptable waiting time for a patient Represent service level agreements for different priority levels for outpatient	Emergency: 30 minutes Urgent: 120 minutes Routine: 240 minutes
condition_type	Determined during the initial referral process Determines: required_resources, wait_time_target, priority_score	Categories: “Emergency”, “Urgent”, “Routine”

2. Participants

The study involved 15 participants, consisting of healthcare administrators and technical staff with varying levels of experience using scheduling systems. This diverse group provided balanced feedback, representing both end-user perspectives and technical insights into system functionality.

3. Testing Scenarios

To evaluate the system thoroughly, participants performed a series of task-based tests simulating real-world scheduling scenarios:

1. Scheduling a New Appointment: Assess task efficiency and accuracy when creating appointments.
2. Rescheduling an Existing Appointment: Evaluate task performance, including time taken and error rates during rescheduling.
3. Handling Overlapping Appointments: Test conflict detection accuracy and evaluate the interface's ability to guide conflict resolution.
4. Viewing and Managing Multi-Doctor Schedules: Assess the system's ability to display and manage complex schedules intuitively.
5. System Performance Under High-Concurrency Scenarios: Measure response times and identify performance degradation when multiple users interact with the system simultaneously.

4. Quantitative Metrics

Quantitative performance metrics were collected during user testing to evaluate system efficiency and technical accuracy:

Metric	Mean Value	Standard Deviation	Observation
Time to Schedule a New Appointment	36 seconds	±8.4 seconds	Efficient for individual tasks but slowed during high-concurrency scenarios.
Time to Reschedule an Appointment	28 seconds	±7.2 seconds	Users desired bulk rescheduling options for efficiency.
Errors Encountered per User	1.2 errors	±0.5 errors	Most errors stemmed from conflicting time slots or misunderstood inputs.
System Response Time	1.5 seconds (average)	±0.6 seconds	Performance degraded during peak load tests.
Conflicts Handled Correctly	92%	±4.3%	Conflict detection was accurate, but resolution guidance was unclear.

5. Qualitative Feedback

Qualitative data were collected through user interviews, post-task surveys, and observations during testing. Feedback was thematically analyzed to identify key areas for improvement:

Category	Summary of User Comments
Ease of Use	"The interface is mostly intuitive, but some advanced options, like recurring scheduling, are hard to locate."
System Intuitiveness	"Navigation is straightforward for basic tasks, but multi-doctor schedules require better visualization."

Error Handling	"Conflict notifications are helpful, but resolution steps need to be more detailed."
Notifications	"The automated reminders are effective, but users want more customizable templates."

Results

Sorting Algorithm:

The sorting algorithm employed in this study demonstrated significant performance across multiple key metrics. It ensured precise prioritization of condition types, adhering strictly to urgency-based scheduling criteria, thereby minimizing errors in task assignment. Designed for immediate decision-making, the algorithm consistently delivered rapid execution times, supporting real-time operational needs. Its performance scaled linearly with increasing dataset sizes, highlighting its capacity to handle larger and more complex data without a degradation in efficiency. Additionally, adherence to prioritization rules maintained fairness in resource allocation by preventing bias, while preserving all input attributes in the output ensured consistency and reliability of the data throughout the processing pipeline.

Performance metrics provide quantitative measures of the algorithm’s effectiveness in meeting its design assumptions. Key metrics include accuracy, execution time, scalability, fairness, and data integrity. These metrics were critical in evaluating the algorithm's capacity to manage healthcare scheduling complexities.

The algorithm's ability to prioritize patients based on urgency, scheduled time, and priority score was confirmed through test outputs. Emergency cases were consistently prioritized over urgent and routine cases, ensuring that critical needs were addressed promptly. This capability demonstrated the algorithm’s accuracy in aligning with healthcare priorities.

Efficiency was evident in the sorting process, as the algorithm handled datasets without significant delays. By employing a stable sorting approach with tuple keys, the algorithm optimized performance, ensuring quick execution for immediate decision-making. Scalability was another key strength. Although the test dataset included a moderate number of records, the algorithm demonstrated linear growth in performance, suggesting its ability to manage larger datasets. Further testing with varying data volumes could validate its robustness.

Fairness in resource allocation is critical in healthcare, and the algorithm’s adherence to condition types and priority scores ensured that patients with greater medical urgency were not deprioritized due to less critical factors. Moreover, data integrity was maintained throughout the process. Test cases confirmed that all input data attributes were preserved in the output, ensuring no loss of critical patient information. Together, these metrics confirm the sorting algorithm’s effectiveness in prioritizing and sorting patient appointments. By balancing in-patient and out-patient scheduling complexities, the algorithm successfully meets its design goals.

Dynamic Programming:

Dynamic programming approaches were evaluated through two optimization strategies: daily and monthly scheduling. Daily optimization excelled in reducing average wait times due to its short-term focus, while monthly optimization achieved balanced scheduling decisions across a broader time horizon. The single-day scope of daily optimization resulted in high room utilization rates, catering to immediate needs. Conversely, monthly optimization distributed resources more evenly over multiple days, offering improved overall balance and adaptability.

Scalability emerged as a critical factor distinguishing the two approaches. Daily optimization was most effective for smaller datasets, where its focus on immediate performance metrics like room utilization and wait time reduction proved advantageous. In contrast, monthly optimization demonstrated superior performance with larger datasets, enabling dynamic adjustments through cost functions to balance room allocation and wait times over extended periods. Flexibility further differentiated the two approaches; daily scheduling was more rigid, while monthly optimization's modular design allowed for greater adaptability to changing requirements.

Reporting capabilities also varied significantly. Daily optimization provided basic performance metrics, sufficient for short-term evaluations. On the other hand, monthly optimization generated detailed tabular reports and comprehensive monthly statistics, offering stakeholders valuable insights for long-term planning.

Key performance metrics reinforced these findings. Average wait times were calculated daily in both approaches, but monthly optimization extended this analysis over longer periods, ensuring greater consistency. Room utilization was another highlight: daily optimization maximize usage within a single day, whereas monthly optimization ensured balanced room usage to address variable demands across multiple days. Lastly, implementation complexity was a differentiating factor. Daily optimization was straightforward and quick to implement, while monthly optimization required a more sophisticated modular design. This added complexity enhanced its adaptability and scalability, making it better suited for long-term, large-scale applications.

Metric	**Approach 1**	**Approach 2**
Average Wait Time	Daily optimization	Monthly optimization
Room Utilization	High for single-day scope	Balanced across multiple days
Scalability	Small datasets	Large datasets
Flexibility in Optimization	Limited	Dynamic with cost functions
Reporting and Insights	Basic	Detailed with tabular reports

Graph-Based Optimization:

The graph-based optimization method yielded transformative results in several critical areas. Patients were prioritized based on condition types and priority scores, ensuring that urgent cases received timely care and significantly decreasing overall wait times. This prioritization strategy directly addressed patient needs, fostering an equitable system that ensured those requiring immediate attention were attended to promptly. As a result, patient satisfaction improved, as urgent medical concerns were addressed without unnecessary delays.

Doctor schedules were optimized by prioritizing efficient doctors with higher performance metrics, maximizing resource utilization and minimizing idle time. This method not only enhanced productivity but also ensured that highly skilled professionals contributed effectively to the overall system, leading to improved care delivery.

Furthermore, the algorithm balanced workloads by factoring in variables such as overtime and rest periods, significantly reducing the risk of burnout and promoting a sustainable work environment for healthcare staff.

Resource utilization also saw marked improvement. By incorporating room cleaning requirements and other operational dependencies, the optimization achieved an efficient balance in facility usage. This comprehensive approach ensured that all available resources, from consultation rooms to medical equipment, were utilized to their fullest potential without overburdening any single aspect of the system.

A notable strength of the graph-based approach was its adaptability. The algorithm's ability to update schedules in real-time ensured responsiveness to dynamic operational conditions, such as last-minute appointment cancellations or unexpected emergencies. This feature enhanced the system's flexibility, allowing it to maintain consistent efficiency even when confronted with unpredictable challenges. Moreover, the implementation of these dynamic adjustments improved coordination across departments, further solidifying the optimization's role in streamlining operations. Together, these advancements underline the graph-based method's effectiveness in addressing the multifaceted demands of healthcare scheduling.

User Research

The user research project was highly successful in meeting its objectives of evaluating and improving the healthcare scheduling system, as evidenced by user feedback and quantitative results. One of the standout successes was the system's robust conflict detection mechanism, achieving a 92% accuracy rate, which users found reliable for identifying overlapping appointments. Additionally, the streamlined interface for basic tasks, such as scheduling and rescheduling, was well-received, with users completing tasks efficiently under normal conditions. Another highlight was the team's comprehensive approach to collecting and analyzing data, which combined statistical rigor with thematic analysis to provide actionable insights.

However, the project also faced challenges. System scalability emerged as a critical limitation, with response times significantly degrading during stress tests involving high-concurrency scenarios. Similarly, users expressed frustration with the lack of actionable guidance in the conflict resolution process, pointing to the need for clearer resolution steps and alternative suggestions. The system's interface for advanced tasks, such as managing multi-doctor schedules or recurring appointments, was another pain point, with users finding these features less intuitive and harder to locate.

Student feedback from the research team reflected both the learning opportunities and difficulties encountered during the project. Students appreciated the chance to engage with real-world challenges, such as designing effective user interfaces and interpreting diverse datasets. The collaborative environment helped develop teamwork and problem-solving skills, but tight deadlines and the technical complexity of scalability improvements were cited as significant hurdles. Overall, the project provided a valuable educational experience, equipping participants with industry-relevant skills while delivering meaningful improvements to the scheduling system.

Discussion

The healthcare scheduling project successfully demonstrated the integration of sorting algorithms, dynamic programming, and graph-based optimization to address real-world scheduling challenges. Students applied these advanced algorithms to minimize patient wait times, optimize doctor workloads, and improve resource utilization, deepening their understanding of theoretical concepts through practical implementation. For example, the Sorting Team's multi-tiered approach ensured that patients were prioritized efficiently, while the Dynamic Programming Team balanced short-term and long-term scheduling through iterative and modular solutions. The Graph Theory Team optimized patient flow with A* pathfinding and dynamic efficiency tracking, showcasing adaptability in managing real-time constraints.

The project also served as an effective platform for enhancing students' technical and collaborative skills. Through hands-on implementation, students gained deeper insights into the practical applications of sorting, DP, and graph theory, reinforcing their algorithmic understanding. Team specialization promoted cross-functional collaboration, mirroring real-world software development processes where effective communication and task delegation are critical for success. The User Research Team played a key role in bridging technical development with user-centered design, gathering critical feedback through surveys and performance testing. Metrics such as 92% conflict detection accuracy validated the system's technical performance, while user feedback emphasized areas requiring refinement, such as conflict resolution workflows and advanced feature usability. This combination of quantitative and qualitative evaluation ensured a balanced assessment of the project's outcomes.

However, the project also uncovered notable areas for improvement. The conflict resolution interface, although effective in detecting issues, lacked actionable resolution guidance, such as step-by-step recommendations or automatic suggestions for alternate time slots. Scalability under stress conditions proved challenging, with response times degrading significantly during high loads, highlighting the need for backend optimizations like load balancing and query caching. Usability concerns were also raised regarding advanced features, such as recurring scheduling and multi-doctor visualization, which users found unintuitive. Enhancements to the user interface, including improved visualization tools and interactive tooltips, could address these challenges. Additionally, user feedback highlighted the need for a bulk rescheduling feature to simplify large-scale updates, such as provider cancellations.

While the project successfully demonstrated the power of algorithmic optimization in healthcare scheduling, it also highlighted the importance of scalability, intuitive design, and user-driven features in achieving real-world practicality. The combination of strong technical achievements and valuable user feedback provided a comprehensive framework for further refinement. Addressing these challenges will ensure the system is both robust and user-friendly, positioning it as a scalable solution for complex healthcare scheduling scenarios. This project ultimately served as a valuable learning experience, equipping students with critical technical skills and preparing them for the challenges of collaborative, industry-aligned software development.

Conclusion

This project demonstrated the significant potential of advanced algorithmic techniques in addressing the intricate challenges of healthcare scheduling. By employing sorting algorithms, dynamic programming approaches, and graph-based optimization, the project tackled critical aspects of patient prioritization, resource allocation, and

workload management. Each method contributed uniquely to the system's overall effectiveness. The sorting algorithm prioritized patients accurately based on urgency, scheduled times, and priority scores, ensuring that critical cases received timely attention. It also maintained fairness in resource allocation and preserved data integrity, making it a reliable component of the scheduling pipeline.

Dynamic programming approaches showcased the adaptability and scalability needed to handle both short-term and long-term scheduling requirements. Daily optimization excelled in reducing average wait times and maximizing immediate resource utilization, while monthly optimization provided a balanced, long-term solution by dynamically distributing resources across a broader time horizon. These methods demonstrated how algorithmic flexibility could meet the varying demands of healthcare operations, catering to small-scale and large-scale environments alike.

The graph-based optimization approach proved particularly transformative, combining efficiency and adaptability. It not only prioritized patients based on urgency and condition types but also optimized doctor schedules by leveraging performance metrics, reducing idle time, and promoting equitable workload distribution. Its ability to integrate real-time adjustments into scheduling operations enhanced system responsiveness to dynamic challenges, such as unexpected emergencies or last-minute cancellations. This adaptability, combined with improved resource utilization through comprehensive modeling of dependencies, ensured a sustainable and efficient operational framework.

From an educational perspective, this project successfully met its objectives of enhancing algorithmic proficiency, fostering teamwork, and providing students with industry-oriented experience. Participants developed and tested advanced algorithms under realistic constraints, gaining invaluable insights into the practical applications of theoretical concepts. Collaborative efforts across specialized teams, such as sorting, dynamic programming, and graph optimization groups, replicated professional environments and emphasized the importance of communication, coordination, and interdisciplinary problem-solving. These skills are essential for addressing complex challenges in real-world scenarios.

The project also highlighted areas for improvement and future exploration. Testing the algorithms with larger, more diverse datasets and introducing more complex real-world constraints, such as evolving patient conditions or sudden resource shortages, could further validate their robustness and scalability. Additionally, integrating advanced predictive analytics or machine learning models could enhance the system’s ability to anticipate demand patterns and optimize resources proactively. This could further refine the scheduling process and provide even more actionable insights for decision-makers.

In conclusion, this project not only provided a platform for students to apply theoretical concepts to practical problems but also underscored the transformative potential of algorithmic approaches in healthcare scheduling. The outcomes highlight the value of combining sorting, dynamic programming, and graph-based methods to address multifaceted operational challenges. By offering a comprehensive framework for patient prioritization, resource allocation, and system adaptability, the project sets a strong foundation for future innovations in this field. Furthermore, the educational impact suggests that similar collaborative projects can significantly enhance both technical skills and teamwork, preparing students for future roles in industry and academia. Future work should focus on incorporating emerging technologies and expanding the scope of applications, ensuring continued advancement in healthcare optimization systems.

References

1. Diestel, R. (2017). Graph Theory (5th ed.). Springer.
2. Dobrzykowski, D., Deilami, V. S., Hong, P., & Kim, S. C. (2015). A structured analysis of operations and supply chain management research in healthcare (1982–2011). Decision Sciences, 28(3), 513-555.
3. Gross, J. L., & Yellen, J. (2005). Graph Theory and Its Applications. CRC Press.
4. Guzzi, P. H., Defilippo, A., & Veltri, P. (2023). A novel network science algorithm for improving triage of patients. In 2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 2803-2808). IEEE.
5. ISO 9241-210:2010. Ergonomics of Human-System Interaction – Part 210: Human-Centred Design for Interactive Systems.
6. Kettinger, W. J., & Grover, V. (1997). The use of computer-mediated communication in an interorganizational context. Decision Sciences, 28(3), 513-555.
7. Nielsen, J. (1994). Usability Engineering. San Francisco: Morgan Kaufmann.
8. Saremi, A., Jula, P., ElMekkawy, T., & Wang, G. G. (2015). Bi-criteria appointment scheduling of patients with heterogeneous service sequences. Expert Systems with Applications, 42(8), 4029-4041.
9. Shneiderman, B., & Plaisant, C. (2010). Designing the User Interface: Strategies for Effective Human-Computer Interaction (5th ed.). Boston: Addison-Wesley.
10. Smith, G., Redfern, O., & Watkinson, P. (2019). The National Early Warning Score 2 (NEWS2): A critical review. Clinical Medicine, 19(3), 260-266.
11. Tullis, T., & Albert, B. (2013). Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics (2nd ed.). Waltham: Morgan Kaufmann.
12. Wuerz, R. C., Milne, L. W., Eitel, D. R., Travers, D., & Gilboy, N. (2000). Reliability and validity of a new five-level triage instrument. Academic Emergency Medicine, 7(3), 236-242.
13. Yinusa, S., & Faezipour, M. (2023). Appointment scheduling model in healthcare using clustering algorithms. arXiv preprint arXiv:1905.03083.
14. Zhu, X., Cai, Q., Wang, J., & Liu, Y. (2020). Identifying critical nodes in hospital networks based on network analysis and data envelopment analysis. International Journal of Environmental Research and Public Health, 17(3), 1008.