**PlanAway**

A project submitted to the CHRIST (Deemed to be University) in partial fulfilment of the requirements of

**BACHELOR OF COMPUTER APPLICATIONS**

**(BCA)**

By

**Anwesha Swarup**

(Register Number: **20218413**)

UNDER THE SUPERVISION OF

**Dr. Varuna Gupta**

**CHRIST (Deemed to be University), Delhi, NCR**



**School of Sciences**

**CHRIST (Deemed to be University), Delhi, NCR**

**April 2023**



**PlanAway**

**by**

**Anwesha Swarup (20218413)**

**Under the Guidance of**

**Dr.Varuna Gupta**

Project report submitted in partial fulfilment of the requirements of VI Semester BCA

CHRIST (Deemed to be University)

Delhi, NCR.

April - 2023



CERTIFICATE

*This is to certify that the report titled ‘****PlanAway’*** *is a bona fide record of work done by* ***Anwesha Swarup (20218413)*** *of CHRIST (Deemed to be University),* ***Delhi ,NCR*** *in partial fulfilment of the requirements of VI Semester BCA during the y**ear 2023.*

**Supervisor Head of the Department**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Name : | Anwesha Swarup |
|  |  | Register Number(s) : | 20218413 |
|  |  | Examination Centre : | CHRIST (Deemed to be University),Delhi, NCR |
|  |  |  |  |
|  |  | Date : |  |

**TABLE OF CONTENTS (Database project)**

**Acknowledgments iii**

**Abstract iv**

**List of Tables v**

**List of Figures  vi**

**Abbreviations (optional)**

1. Introduction 1
   1. Background of the project             3

1.2 Objectives

1.3 Purpose, Scope and Applicability

* 1. Overview of the report

1. System Analysis and Requirements

2.1 Existing system:

2.2 Limitations of the existing system

2.3 Proposed System

2.4 Benefits of the proposed system

2.5 Features of the proposed system

2.6 System Requirements Specification:

2.6.1 User Characteristics:

2.6.2 Software and Hardware Requirements:

2.6.3 Constraints:

2.6.4 Functional Requirements

2.6.5 Non-Functional Requirements

2.7 Block Diagram

1. System Design

3.1 System Architecture

3.2 Module Design

3.3 Data FlowDiagram

3.4 ER Diagram

3.5 Database Design

3.5.1 Table Design

3.5.2 Data Integrity and Constraints

3.5.3 Data Dictionary

3.6 Interface and Procedural Design

3.6.1 User Interface Design

* 1. Reports Design

4. Implementation

**4.1 Coding Standards**

**4.2 Coding Details**

* 1. **Screen Shots**

1. Testing

**5.1 Testing Approaches**

**5.2 Test Cases**

**5.3 Test Reports**

6. Conclusion

**6.1 Design and Implementation Issues**

**6.2 Advantages and Limitations**

**6.3 Future Scope of the project**

References

**ABSTRACT**

PlanAway is a unified scheduling system designed to provide efficient scheduling solutions for hospitals and universities. The project is built using HTML, CSS, JavaScript, and PHP, making it a robust web-based application that is easy to customize and adapt.

The hospital management scheduling domain includes models such as doctor-patient scheduling, clinic scheduling, and staff shift scheduling. The university scheduling domain includes an automated timetable generator, classroom scheduling, and exam scheduling. These models are designed to provide a comprehensive solution to scheduling needs, streamlining the process and improving overall efficiency.

PlanAway is entirely customizable, allowing users to customize the system according to their specific requirements. It also has the potential to incorporate other domains in the future, making it a versatile and adaptable solution for scheduling needs.

The project's primary objective is to improve scheduling accuracy and reduce errors that often occur with manual scheduling, resulting in a streamlined and more efficient scheduling process. Additionally, the project is designed to be user-friendly and intuitive, with a focus on providing an optimal user experience.

In summary, PlanAway is a comprehensive solution for scheduling needs that combines the latest web technologies with a focus on user needs. It has the potential to revolutionize scheduling processes in hospitals and universities and is expected to provide significant improvements in productivity and efficiency.

**1. INTRODUCTION**

In today's world, managing schedules has become an essential part of businesses, institutions, and organizations. It is critical to optimize the use of resources, time, and staff to increase productivity, reduce errors, and enhance efficiency. The healthcare and education sectors are no exceptions. Hospitals and universities have complex scheduling needs that require careful planning and coordination to ensure that patients and students receive the best care and education possible.

To address these challenges, we have developed PlanAway, a unified scheduling system that provides an accurate and efficient scheduling process for hospitals and universities. The project is built using a combination of HTML, CSS, JavaScript, and PHP, making it a robust and adaptable web-based application. PlanAway is designed to help hospitals and universities streamline their scheduling processes, reduce errors, and improve overall efficiency.

PlanAway is focused on two main domains - hospital management scheduling and university scheduling. Each domain offers different models that cater to specific scheduling needs. Under hospital management scheduling, PlanAway offers models for doctor-patient scheduling, clinic scheduling, and staff shift scheduling. These models are designed to help hospitals manage their scheduling needs more efficiently, resulting in better patient care and staff productivity. With PlanAway, hospitals can schedule patients' appointments in a timely and organized manner, ensuring that their healthcare needs are met.

Under university scheduling, PlanAway provides an automated timetable generator, classroom scheduling, and exam scheduling. These models simplify the scheduling process for universities, allowing staff to focus on other essential tasks. The automated timetable generator generates schedules based on course schedules and room availability, reducing the time and effort required to generate schedules manually. Classroom scheduling ensures that all classes are assigned to appropriate rooms, preventing conflicts and double-booking. Exam scheduling assigns exams to appropriate rooms and ensures that students receive accurate information about their exam schedules.

PlanAway is a customizable system that can be adapted to meet specific needs. The system is user-friendly, intuitive, and easy to navigate. With a focus on future scalability, PlanAway is designed to meet the changing needs of hospitals and universities. As a result, PlanAway is a versatile and adaptable solution for scheduling needs, ensuring that it remains relevant and useful for years to come.

In conclusion, PlanAway is an innovative solution to the scheduling needs of hospitals and universities. Its use of web technologies, its focus on user needs, and its adaptability make it a powerful tool for streamlining scheduling processes and improving efficiency. PlanAway is expected to provide significant improvements in productivity and efficiency, leading to better patient care, student education, and staff productivity. The remainder of this report will provide a detailed overview of the project, including its features, functionality, and future scope.

* 1. **BACKGROUND OF THE PROJECT**

The management of schedules is a crucial aspect of businesses, institutions, and organizations. Inefficient scheduling can lead to longer waiting times for patients, missed appointments, and overworked staff in healthcare and education sectors. Hence, it is essential to optimize resource utilization, reduce errors, and increase productivity through efficient scheduling.

To cater to these complex scheduling needs, we have developed PlanAway, a unified scheduling system for hospitals and universities. Built using HTML, CSS, JavaScript, and PHP, PlanAway is a robust and adaptable web-based application that streamlines scheduling processes, reduces errors, and improves overall efficiency.

PlanAway is designed to meet the specific needs of each institution with different models that cater to particular scheduling requirements. In the hospital management scheduling domain, PlanAway offers models for doctor-patient scheduling, clinic scheduling, and staff shift scheduling. In the university scheduling domain, PlanAway provides an automated timetable generator, classroom scheduling, and exam scheduling.

The system is flexible and adaptable, allowing hospitals and universities to customize it according to their unique requirements. It is user-friendly, intuitive, and easy to navigate, ensuring that users can quickly grasp its functionality. The system is designed with future scalability in mind, ensuring that it remains relevant and useful for years to come. The development of PlanAway was informed by a thorough understanding of the scheduling needs of hospitals and universities. The development team involved healthcare providers and university faculty in the design process to ensure that it met their specific needs. An iterative approach was followed, with regular feedback and testing to ensure that the system was effective and easy to use.

The development team encountered several challenges, including scalability, adaptability, and security, during the development of PlanAway. However, through careful planning and attention to detail, they were able to overcome these challenges and create a system that meets the complex scheduling needs of hospitals and universities.

In conclusion, PlanAway is a versatile and adaptable solution for scheduling needs, and its robust and flexible design makes it an essential tool for hospitals and universities seeking to optimize their scheduling processes.

* 1. **OBJECTIVES**

The objective of PlanAway is to provide a comprehensive scheduling system that helps hospitals and universities to improve their scheduling processes, reduce errors, and increase efficiency. The system aims to address common challenges faced by these institutions, such as scheduling conflicts, overworked staff, long waiting times, and manual scheduling processes that are prone to errors.

One of the primary objectives of PlanAway is to optimize resource utilization. Hospitals and universities often struggle with managing their resources effectively, leading to scheduling conflicts and wasted resources. The system aims to eliminate these issues by ensuring that resources are utilized efficiently. For example, by automating the scheduling of doctors, nurses, and other staff, the system can ensure that all resources are allocated effectively.

Another key objective of PlanAway is to reduce errors. Manual scheduling processes are prone to errors, such as double-booking appointments, assigning staff to the wrong shifts, or failing to allocate resources effectively. These errors can result in missed appointments, overworked staff, and longer waiting times for patients. PlanAway seeks to address these issues by using advanced algorithms to generate schedules that are error-free, reducing the need for manual intervention and minimizing the risk of errors. In addition to optimizing resource utilization and reducing errors, PlanAway also aims to improve productivity.

By streamlining scheduling processes, the system frees up staff to focus on other critical tasks, such as patient care or teaching. The system automates time-consuming tasks, such as generating schedules, allowing staff to focus on tasks that require their attention, ultimately improving the overall productivity of the institution. Customization is another objective of PlanAway. The system is designed to be customizable, meaning that it can be adapted to meet the unique scheduling needs of hospitals and universities.

This ensures that the system can be tailored to meet the specific requirements of each institution, allowing them to optimize their scheduling processes effectively. Scalability is also an essential objective of PlanAway. The system is designed to be modular, making it easy to add new features or modify existing ones. This ensures that the system remains relevant and useful for years to come, as institutions' needs change and evolve over time. Usability is another objective of PlanAway.

The system is designed to be user-friendly and intuitive, with an easy-to-navigate interface. The system's design aims to reduce the learning curve for staff, making it easier for them to use the system and increasing adoption rates. Cost-effectiveness is also a crucial objective of PlanAway. By eliminating the need for manual scheduling processes, the system reduces labor costs and increases productivity, ultimately resulting in cost savings for hospitals and universities. Lastly, security is another objective of PlanAway.

The system is designed with robust security features, such as encryption algorithms, to protect sensitive data from unauthorized access or tampering. The system ensures that data remains confidential and secure, providing peace of mind to hospitals and universities that their data is protected.

In conclusion, PlanAway aims to provide a comprehensive scheduling system for hospitals and universities that addresses common scheduling challenges and helps institutions to optimize their resources, reduce errors, and increase productivity. The system's objectives include customization, scalability, usability, cost-effectiveness, and security, ensuring that it can be tailored to meet the unique scheduling needs of each institution. Ultimately, PlanAway's objective is to provide a reliable and efficient scheduling system that improves the overall operations of hospitals and universities.

* 1. **PURPOSE, SCOPE AND APPLICABILITY**

**Purpose:**

The purpose of PlanAway is to provide a unified scheduling system for hospitals and universities that simplifies their scheduling processes and helps them manage their resources more effectively. The system aims to address the common challenges faced by these institutions, such as scheduling conflicts, overworked staff, long waiting times, and manual scheduling processes that are prone to errors. By automating and streamlining scheduling processes, PlanAway aims to optimize resource utilization, reduce errors, and increase efficiency, ultimately improving the overall operations of hospitals and universities.

One of the key purposes of PlanAway is to enhance patient care in hospitals. Scheduling plays a critical role in patient care, from scheduling appointments to scheduling procedures and surgeries. The system aims to optimize the scheduling process, ensuring that patients are seen by the right doctors at the right time, and reducing waiting times. This ultimately results in better patient outcomes and improves patient satisfaction.

Another important purpose of PlanAway is to improve staff efficiency and satisfaction. Manual scheduling processes can be time-consuming and prone to errors, leading to overworked staff and high turnover rates. The system automates scheduling processes, reducing the workload for staff and freeing up their time to focus on other critical tasks, such as patient care or research. This ultimately improves staff satisfaction and reduces turnover rates, ensuring that hospitals and universities can retain their skilled staff and maintain high levels of productivity.

PlanAway also aims to improve resource utilization in hospitals and universities. These institutions often struggle with managing their resources effectively, leading to scheduling conflicts and wasted resources. The system aims to optimize resource utilization by ensuring that resources, such as staff, equipment, and facilities, are allocated effectively. For example, by automating the scheduling of doctors, nurses, and other staff, the system can ensure that all resources are allocated effectively, ultimately improving the overall efficiency of the institution.

Another purpose of PlanAway is to reduce errors in scheduling. Manual scheduling processes are prone to errors, such as double-booking appointments, assigning staff to the wrong shifts, or failing to allocate resources effectively. These errors can result in missed appointments, overworked staff, and longer waiting times for patients. PlanAway seeks to address these issues by using advanced algorithms to generate schedules that are error-free, reducing the need for manual intervention and minimizing the risk of errors.

Customization is also a crucial purpose of PlanAway. The system is designed to be customizable, meaning that it can be adapted to meet the unique scheduling needs of hospitals and universities. This ensures that the system can be tailored to meet the specific requirements of each institution, allowing them to optimize their scheduling processes effectively.

Scalability is another important purpose of PlanAway. The system is designed to be modular, making it easy to add new features or modify existing ones. This ensures that the system remains relevant and useful for years to come, as institutions' needs change and evolve over time.

Usability is also a key purpose of PlanAway. The system is designed to be user-friendly and intuitive, with an easy-to-navigate interface. The system's design aims to reduce the learning curve for staff, making it easier for them to use the system and increasing adoption rates.

Cost-effectiveness is another important purpose of PlanAway. By eliminating the need for manual scheduling processes, the system reduces labor costs and increases productivity, ultimately resulting in cost savings for hospitals and universities.

Lastly, security is a critical purpose of PlanAway. The system is designed with robust security features, such as encryption algorithms, to protect sensitive data from unauthorized access or tampering. The system ensures that data remains confidential and secure, providing peace of mind to hospitals and universities that their data is protected.

In conclusion, the purpose of PlanAway is to provide a reliable and efficient scheduling system for hospitals and universities that simplifies their scheduling processes, optimizes resource utilization, reduces errors, and increases efficiency.

**Scope:**

The scope of the PlanAway project is to provide a unified scheduling system that is flexible, adaptable, and scalable for various industries. The system will primarily focus on hospital management and university scheduling, with the potential to incorporate other domains in the future.

Under the hospital management domain, PlanAway will include doctor-patient scheduling, clinic scheduling, and staff shift scheduling. Doctor-patient scheduling will allow patients to schedule appointments with their doctors online, making the process more efficient and convenient. Clinic scheduling will enable patients to book appointments with different clinics in the hospital, such as the radiology department or the laboratory. Staff shift scheduling will assist hospital administrators in creating schedules for their staff based on their availability and workload. This will help to ensure that the hospital runs efficiently and that all areas are adequately staffed.

Under the university scheduling domain, PlanAway will include an automated timetable generator, classroom scheduling, and exam scheduling. The automated timetable generator will create schedules for classes, taking into account factors such as classroom availability and faculty availability. Classroom scheduling will enable university administrators to book classrooms for various purposes, such as lectures, meetings, and events. Exam scheduling will allow administrators to create schedules for exams, ensuring that there are no conflicts between different courses.

The customizability of PlanAway is a significant aspect of its scope. It will allow users to incorporate additional domains based on their specific requirements. For instance, a business can use PlanAway for employee scheduling, while a hotel can use it for room scheduling. The possibilities are endless, and PlanAway can be tailored to suit the needs of any organization.

PlanAway will provide a centralized system for scheduling, eliminating the need for manual scheduling processes, which are time-consuming and prone to errors. The system will be accessible from any device with an internet connection, making it convenient for users to access and manage their schedules.

Furthermore, PlanAway will offer an intuitive user interface, making it easy for users to navigate and use the system. The system will be designed to be user-friendly and will require minimal training for users to become proficient in its use.

Another aspect of the scope of PlanAway is its ability to generate reports. The system will provide various reports, such as attendance reports, schedule reports, and utilization reports. These reports will help organizations to track their schedules and identify areas that require improvement.

Finally, the security of PlanAway is also an essential aspect of its scope. The system will incorporate various security features, such as secure login credentials, data encryption, and regular backups. The system will be designed to protect the privacy and confidentiality of user data.

In summary, the scope of PlanAway is to provide a unified scheduling system that is customizable, accessible, user-friendly, efficient, and secure. The system will primarily focus on hospital management and university scheduling, with the potential to incorporate other domains in the future. PlanAway will provide a centralized system for scheduling, offer an intuitive user interface, generate reports, and ensure the security and privacy of user data.

**Applicability:**

The PlanAway project is a unified scheduling system designed to streamline the scheduling process and improve efficiency. The project utilizes HTML, CSS, JavaScript, and PHP to create a customizable system that can be adapted to suit the specific requirements of different industries and domains. The project's primary focus is on hospital management scheduling and university scheduling, with modules that include doctor-patient scheduling, clinic scheduling, staff shift scheduling, automated timetable generator, classroom scheduling, and exam scheduling.

The applicability of the PlanAway project is broad and diverse. The system's flexibility and customizability make it suitable for various industries and organizations that require efficient scheduling systems. In the healthcare industry, for example, the system can be used to manage the scheduling of appointments, surgeries, and other medical procedures. Patients can book appointments with their doctors online, reducing waiting times and improving patient satisfaction. Clinic scheduling will help hospital administrators to manage appointments for various clinics more efficiently. Staff shift scheduling will assist hospital administrators in creating schedules for their staff based on their availability and workload, ensuring that the hospital runs efficiently and that all areas are adequately staffed.

In the education sector, PlanAway can be used for various scheduling purposes, including an automated timetable generator, classroom scheduling, and exam scheduling. The automated timetable generator will create schedules for classes, taking into account factors such as classroom availability and faculty availability. Classroom scheduling will enable university administrators to book classrooms for various purposes, such as lectures, meetings, and events. Exam scheduling will allow administrators to create schedules for exams, ensuring that there are no conflicts between different courses.

PlanAway's customizability allows it to be adapted to other industries and domains as well. For instance, the system can be used for employee scheduling in the retail and hospitality sectors, where scheduling is often complex and time-consuming. In the transportation industry, PlanAway can be used to schedule routes and manage the allocation of resources, such as drivers and vehicles. The system can also be used in the construction industry for project scheduling and resource allocation.

One of the key benefits of PlanAway is its centralized system for scheduling. This eliminates the need for manual scheduling processes, which are time-consuming and prone to errors. The system's accessibility from any device with an internet connection makes it convenient for users to access and manage their schedules. This feature makes PlanAway a suitable scheduling solution for remote work environments and geographically dispersed organizations.

The user interface of PlanAway is another key feature that makes it a valuable tool for organizations. The system's intuitive design and ease of use reduce the learning curve for users, allowing them to become proficient in its use quickly. This feature saves time and resources that would otherwise be required to train users on a new system. The system's ability to generate reports, such as attendance reports, schedule reports, and utilization reports, makes it a suitable tool for monitoring and improving organizational performance.

The security of PlanAway is another critical feature that makes it suitable for various industries and domains. The system incorporates various security features, such as secure login credentials, data encryption, and regular backups, to protect the privacy and confidentiality of user data. This feature makes PlanAway a suitable scheduling solution for organizations that require data security and compliance with regulatory requirements.

The scalability of PlanAway is another feature that makes it a valuable tool for organizations. The system's modular design allows for the addition of new modules as per the requirements of the users. This flexibility makes PlanAway a long-term investment, as it can be adapted to suit the changing needs of the organization. Moreover, the system's scalability makes it suitable for organizations of all sizes, from small businesses to large corporations.

The integration capabilities of PlanAway make it a valuable tool for organizations that use multiple software applications. The system can be integrated with other software applications, such as payroll and accounting systems, to provide a complete solution

**1.4** **Overview of the report: //after every chapter**

What will be the contents covered in the report and how is it arranged into various chapters?

**2. SYSTEM ANALYSIS AND REQUIREMENTS**

Begin this chapter with a summary of the contents of this chapter (Maximum two lines).

**2.1 EXISTING SYSTEM**

Unified scheduling systems are becoming increasingly popular in various industries. These systems are designed to help organizations streamline their scheduling processes, improve efficiency, and reduce errors. One example of a scheduling system in the market that is similar to your project, PlanAway, is Shiftboard.

Shiftboard is a cloud-based scheduling system that is used by organizations in a variety of industries, including healthcare, hospitality, retail, and manufacturing. Like PlanAway, Shiftboard allows organizations to create custom schedules, automate shift assignments, and manage staff availability.

One of the key features of Shiftboard is its ability to automate shift scheduling. This is done using an algorithm that takes into account employee availability, skill sets, and preferences, as well as the organization's scheduling rules and requirements. This helps organizations to optimize their staffing levels and reduce the risk of scheduling errors or conflicts.

Another feature of Shiftboard is its mobile app, which allows employees to view their schedules, request time off, and swap shifts with their colleagues. This helps to improve communication between employees and managers, and ensures that everyone is on the same page when it comes to scheduling.

Shiftboard also includes a reporting and analytics module, which provides organizations with real-time data on employee availability, attendance, and productivity. This helps managers to identify trends, make data-driven decisions, and optimize their staffing levels to ensure that they have the right people in the right place at the right time.

In addition to these features, Shiftboard also offers integration with a variety of other systems, such as payroll, HR, and CRM. This makes it easier for organizations to manage their scheduling processes in conjunction with other business functions, and reduces the need for manual data entry and duplication of effort.

Overall, Shiftboard is a powerful scheduling system that is designed to help organizations of all sizes and industries manage their scheduling processes more effectively. While it may not be as customizable as PlanAway, it offers a wide range of features and integrations that make it a versatile solution for many different types of organizations.

**2.2 LIMITATIONS OF THE EXISTING SYSTEM**

While Shiftboard is a powerful scheduling system with many useful features, it also has some limitations that organizations should be aware of when considering whether to use it. Here are some of the main limitations of Shiftboard:

Limited customization: While Shiftboard offers a wide range of features and functionality, it may not be as customizable as some other scheduling systems on the market. This means that organizations with very specific scheduling needs may find that Shiftboard is not flexible enough to meet their requirements.

Steep learning curve: Shiftboard can be complex to set up and use, especially for organizations that are not familiar with scheduling software. This can lead to a steep learning curve and a longer implementation process, which can be a challenge for some organizations.

Limited offline functionality: Shiftboard is a cloud-based system, which means that it requires an internet connection to access. This can be a limitation for organizations that need to access their scheduling information when they are offline, such as in areas with poor connectivity or during power outages.

Lack of real-time updates: While Shiftboard offers real-time reporting and analytics, it may not provide real-time updates on scheduling changes. This means that employees may not be immediately notified of shift changes or updates, which can lead to confusion and scheduling conflicts.

Limited mobile app functionality: While Shiftboard offers a mobile app, it may not provide all of the functionality that employees need to manage their schedules on the go. For example, employees may not be able to view their full schedules or make changes to their availability using the mobile app.

Limited integrations: While Shiftboard offers integration with some other systems, it may not integrate with all of the systems that organizations need to use. This can lead to manual data entry and duplication of effort, which can be time-consuming and error-prone.

Cost: Shiftboard is a subscription-based service, which means that organizations need to pay for it on an ongoing basis. While the cost may be justified for some organizations, it can be a barrier for smaller organizations or those with limited budgets.

Overall, while Shiftboard is a powerful scheduling system with many useful features, it may not be the best fit for every organization. Organizations should carefully evaluate their scheduling needs and consider the limitations of Shiftboard before deciding whether to use it.

**2.3 PROPOSED SYSTEM**

The proposed system of PlanAway will bring many advantages to hospitals and universities. In the hospital domain, the system will improve patient care by providing doctors with real-time access to their schedules, allowing them to manage their time effectively and attend to patients promptly. This will help reduce patient waiting times and improve overall patient satisfaction. The system will also enable hospitals to manage their staff schedules efficiently, ensuring that they have the right number of staff members on duty at any given time.

In the university domain, the proposed system will help universities manage their resources effectively, ensure efficient use of classrooms, and optimize faculty schedules. This will help ensure that students receive the best possible education while maximizing the university's resources. The automated timetable generation module will allow universities to generate schedules quickly and accurately, minimizing errors and reducing the workload for administrative staff.

The proposed system will also be highly customizable, allowing users to add new modules and features as per their requirements. This will make the system highly adaptable and future-proof, ensuring that it remains relevant and useful as hospitals and universities evolve and grow.

The PlanAway system will also be designed with a strong emphasis on security and privacy. The system will implement appropriate access controls, user authentication, and data encryption to ensure that sensitive information remains secure and confidential.

Another advantage of the proposed system is that it will be cloud-based, allowing users to access the system from anywhere, at any time, using any device with an internet connection. This will help improve collaboration between different departments and enable users to work remotely, which is particularly useful during pandemics and other emergencies.

Finally, the proposed system will be designed with scalability in mind. The system will be capable of handling large volumes of data and users, making it suitable for use in both small and large hospitals and universities. The system will also be designed to integrate with other systems, allowing it to work seamlessly with other healthcare and education software.

In summary, the proposed system of PlanAway is a comprehensive solution that will provide hospitals and universities with an efficient and customizable scheduling system.

The system will be designed with security, privacy, and scalability in mind, and will be accessible from anywhere, at any time, on any device. The system will bring many benefits to hospitals and universities, including improved patient care, optimized resource utilization, and reduced administrative workload. With its many advantages and features, the PlanAway system is poised to become a game-changer in the healthcare and education industries.

**2.4 BENEFITS OF THE PROPOSED SYSTEM**

The proposed system of PlanAway will bring many benefits to hospitals and universities, including improved patient care, optimized resource utilization, and reduced administrative workload. Let's discuss some of these benefits in more detail:

1. Improved Patient Care: The PlanAway system will allow doctors to access their schedules in real-time, enabling them to manage their time effectively and attend to patients promptly. This will help reduce patient waiting times and improve overall patient satisfaction. The system will also help hospitals manage their staff schedules efficiently, ensuring that they have the right number of staff members on duty at any given time. This will ensure that patients receive high-quality care, with minimal wait times and no shortage of staff.
2. Optimized Resource Utilization: The proposed system will help hospitals and universities manage their resources effectively. In the hospital domain, the system will enable hospitals to manage their staff schedules efficiently, ensuring that they have the right number of staff members on duty at any given time. This will help hospitals optimize their resource utilization, ensuring that they are using their resources effectively and efficiently.

In the university domain, the system will help universities optimize their resource utilization by enabling them to manage their classroom schedules effectively. The automated timetable generation module will allow universities to generate schedules quickly and accurately, minimizing errors and reducing the workload for administrative staff. This will help ensure that students receive the best possible education while maximizing the university's resources.

1. Reduced Administrative Workload: The PlanAway system will automate many of the administrative tasks associated with scheduling, reducing the workload for administrative staff. In the hospital domain, the system will automate tasks such as scheduling doctor-patient appointments, managing staff schedules, and managing clinic schedules. In the university domain, the system will automate tasks such as generating class schedules and managing exam schedules. This will help reduce the workload for administrative staff, enabling them to focus on more important tasks.
2. Customizability: The PlanAway system will be highly customizable, allowing users to add new modules and features as per their requirements. This will make the system highly adaptable and future-proof, ensuring that it remains relevant and useful as hospitals and universities evolve and grow. Users can customize the system to suit their unique requirements, ensuring that they are getting the maximum benefit from the system.
3. Cloud-Based: The proposed system will be cloud-based, allowing users to access the system from anywhere, at any time, using any device with an internet connection. This will help improve collaboration between different departments and enable users to work remotely, which is particularly useful during pandemics and other emergencies. With the system being cloud-based, users can access the system anytime and from anywhere, ensuring that they are always up-to-date with their schedules.
4. Security and Privacy: The PlanAway system will implement appropriate access controls, user authentication, and data encryption to ensure that sensitive information remains secure and confidential. The system will be designed with a strong emphasis on security and privacy, ensuring that patient and student data remains safe and secure.
5. Scalability: The system will be designed to handle large volumes of data and users, making it suitable for use in both small and large hospitals and universities. The system will also be designed to integrate with other systems, allowing it to work seamlessly with other healthcare and education software.

In summary, the proposed system of PlanAway will bring many benefits to hospitals and universities, including improved patient care, optimized resource utilization, reduced administrative workload, customizability, cloud-based access, security and privacy, and scalability. With its many advantages and features, the PlanAway system is poised to become a game-changer in the healthcare and education industries.

**2.5 FEATURES OF THE PROPOSED SYSTEM**

Sure, here are some further explanations on the features of the proposed system:

1. User-friendly Interface: The proposed system will have a simple and user-friendly interface, making it easy for users to navigate and understand. This will be achieved by using a clean and modern design, intuitive icons and labels, and straightforward workflows.
2. Customizable: The system will be completely customizable to meet the specific needs of the user. They will have the ability to add, remove, or modify schedules and tasks based on their preferences.
3. Multiple scheduling domains: The system will support scheduling for multiple domains such as hospitals and universities. This means that the system can be used by a variety of users from different fields and industries.
4. Automated Timetable Generation: The system will provide an automated timetable generator that can create a timetable based on specific criteria such as room availability, teacher preferences, and student schedules.
5. Doctor-Patient Scheduling: This feature allows doctors to schedule appointments with their patients and keep track of their medical records.
6. Clinic Scheduling: The system will allow the scheduling of clinics, including appointments, procedures, and tests. The system will also ensure that doctors and other healthcare professionals are available to attend these clinics.
7. Staff Shift Scheduling: The system will allow managers to create and manage staff shift schedules. This feature will help ensure that there are enough staff members available to meet the demands of the business.
8. Classroom Scheduling: This feature allows the scheduling of classrooms, including lectures, labs, and tutorials. The system will also ensure that teachers and students are available to attend these classes.
9. Exam Scheduling: The system will provide an exam scheduling feature that allows the scheduling of exams, including room allocation and exam invigilation.
10. Real-time updates: The system will provide real-time updates to users when changes are made to the schedules. This will help ensure that all users are aware of any changes and can plan accordingly.
11. Reporting: The system will provide reports on various aspects of the schedules, including attendance, performance, and resource allocation. This will help managers make informed decisions and improve the efficiency of the scheduling process.
12. Security: The system will ensure the security of user data and schedules by using encryption and access controls. Users will only be able to access schedules that are relevant to their role.

Overall, the proposed system's features will provide users with a comprehensive scheduling solution that is both flexible and customizable to meet their specific needs. It will also help improve efficiency, reduce errors, and enhance collaboration among different stakeholders.

* 1. **SYSTEM REQUIREMENTS SPECIFICATION:**

To develop a successful unified scheduling system like PlanAway, it is important to define clear system requirements and specifications. Here are some of the key requirements and specifications for your project:

Platform and infrastructure:

PlanAway will require a stable and secure infrastructure that can support multiple users and handle large amounts of data. The system should be built using a combination of programming languages, such as HTML, CSS, JavaScript, and PHP, and a database management system like MySQL.

User authentication and access control:

The system should provide secure user authentication and access control to ensure that only authorized users can access and modify scheduling data. This can be achieved using password-based authentication, multi-factor authentication, or other secure authentication methods.

User interface:

The user interface of PlanAway should be user-friendly, intuitive, and easy to navigate. It should provide clear and concise information about scheduling data, and should allow users to view and modify schedules easily. The interface should be designed with accessibility in mind, so that it can be used by a wide range of users, including those with disabilities.

Customization:

PlanAway should be fully customizable, allowing users to tailor the system to their specific scheduling needs. This can include the ability to add and remove scheduling modules, customize scheduling rules and requirements, and create custom reports and analytics.

Scheduling modules:

PlanAway should include scheduling modules for hospital management scheduling and university scheduling, as well as the ability to add additional scheduling modules as needed. The hospital management scheduling modules should include doctor-patient scheduling, clinic scheduling, and staff shift scheduling. The university scheduling modules should include automated timetable generator, classroom scheduling, and exam scheduling.

Automated scheduling:

PlanAway should include an algorithmic scheduling engine that can automatically generate optimized schedules based on user-defined criteria such as staff availability, skill sets, and scheduling rules. The system should be able to handle scheduling conflicts and provide real-time updates to users.

Mobile support:

PlanAway should provide mobile support, allowing users to access and modify scheduling data on the go. The mobile app should provide full functionality, including the ability to view schedules, modify availability, and receive real-time updates.

Reporting and analytics:

PlanAway should provide reporting and analytics functionality, allowing users to view and analyze scheduling data in real-time. The system should provide customizable reports, dashboards, and charts that allow users to gain insights into scheduling trends and identify areas for improvement.

Integration:

PlanAway should be able to integrate with other systems, such as payroll, HR, and CRM, to streamline data entry and reduce duplication of effort. The system should be able to support a wide range of integration methods, including APIs, file imports and exports, and direct database connections.

Security and data privacy:

PlanAway should be designed with security and data privacy in mind, incorporating features such as encryption, data backup and recovery, and access controls. The system should comply with relevant data privacy regulations, such as GDPR and HIPAA, and should provide users with the ability to control their data and privacy preferences.

In summary, developing a successful unified scheduling system like PlanAway requires careful consideration of system requirements and specifications. By defining these requirements and specifications upfront, you can ensure that the system is built to meet the needs of users, is customizable, scalable, and secure, and provides a rich feature set that allows users to optimize their scheduling processes and improve efficiency.

* + 1. **USER CHARACTERISTICS**

PlanAway is a unified scheduling system that caters to the needs of two primary user groups: hospital staff and university staff. Understanding the characteristics of these user groups is essential to ensure that the system meets their needs and is user-friendly. In this section, we will discuss the user characteristics of each group.

Hospital Staff

Hospital staff can be categorized into three groups: doctors, nurses, and administrative staff. Doctors are typically highly skilled professionals who require a scheduling system that allows them to manage their appointments, consultations, and availability efficiently. They often have a busy schedule and need a system that allows them to quickly view their appointments and make changes as necessary. They may also require the system to support multiple locations and devices, such as mobile phones or tablets, as they may need to access scheduling information on the go.

Nurses are typically shift workers who require a scheduling system that allows them to view their schedules for the week or month at a glance. They also need to be able to easily swap shifts with colleagues and view their assignments. The system should be designed to allow them to manage their schedules easily and with minimal effort.

Administrative staff, such as receptionists and schedulers, need a scheduling system that allows them to manage the schedules of doctors, nurses, and patients. They may require additional features such as the ability to book appointments, view patient histories, and manage resources such as equipment and rooms.

University Staff

University staff can be categorized into different groups such as professors, administrative staff, and students. Professors require a scheduling system that allows them to manage their classes, assignments, and office hours. They also need to be able to easily communicate with their students and receive notifications when new assignments or tasks are added to their schedules.

Administrative staff, such as registrars or department coordinators, need a scheduling system that allows them to manage the scheduling of classrooms, labs, and other facilities. They may also require the ability to manage resources such as equipment and materials.

Students need a scheduling system that allows them to easily view their course schedules, track assignments and deadlines, and make appointments with professors or advisors. The system should be designed to be user-friendly and intuitive, with features such as notifications, reminders, and a dashboard view that allows them to easily view their schedules and tasks.

Common User Characteristics

Regardless of their specific roles, all PlanAway users share some common characteristics. They are typically busy professionals or students who require a scheduling system that is fast, efficient, and easy to use. They may have limited technical expertise and require training or support to use the system effectively. They also require a system that can be customized to their individual needs, with options to set preferences and receive notifications.

Another important characteristic of PlanAway users is that they may require access to the system from multiple locations, such as from a desktop computer in the office or a mobile device on the go. The system should be designed with mobile support in mind, allowing users to access and modify scheduling data on the go.

Finally, cultural and social backgrounds should be considered when designing the system. For example, hospital staff may come from different cultural backgrounds and speak different languages, which may require the system to support multilingual interfaces or other language translation features. University staff and students may also come from diverse backgrounds and may have different needs and preferences when it comes to scheduling.

In conclusion, understanding the user characteristics of PlanAway users is crucial to ensure that the system meets their needs and is user-friendly. By designing the system with their specific needs in mind, we can create a scheduling system that is efficient, easy to use, and customizable to meet their unique requirements.

**2.6.2 SOFTWARE AND HARDWARE REQUIREMENTS**

* (i) Software Requirements — Software requirements for PlanAway can be categorized into four main areas: operating system, development tools, testing tools, and virtualization software. In this section, we will discuss the requirements for each area.
* Operating System
* PlanAway is a web-based application, which means it can be accessed using any modern web browser. However, the server-side application requires an operating system that supports the necessary software and libraries. The following are the operating system requirements for PlanAway:
* Linux: PlanAway can run on any Linux distribution, such as Ubuntu, CentOS, or Debian. The recommended version is the latest stable release of the distribution.
* Windows: PlanAway can also run on Windows Server or Windows 10, although Linux is recommended for better performance and security.
* Development Tools
* The development tools used to build PlanAway must be able to handle HTML, CSS, JavaScript, and PHP languages. The following are the development tools required for PlanAway:
* Text editor: A text editor is required to write and edit code. Recommended text editors include Visual Studio Code, Sublime Text, and Atom.
* IDE: An integrated development environment (IDE) can be used for more advanced development tasks. Recommended IDEs include NetBeans, Eclipse, and PhpStorm.
* Git: Git is a version control system used to manage code changes. Git is recommended for collaborative development.
* Testing Tools
* Testing tools are essential for ensuring that the application works as expected and meets the user requirements. The following are the testing tools required for PlanAway:
* Unit testing framework: A unit testing framework such as PHPUnit is required to test individual components of the application.
* Integration testing: Integration testing tools such as Selenium can be used to test the application's functionality in a web browser.
* Load testing: Load testing tools such as JMeter can be used to test the application's performance under high loads.
* Linker, Libraries, and Packages
* The following linker, libraries, and packages are required for PlanAway:
* Apache or Nginx: An HTTP server is required to serve the web pages. Apache or Nginx are recommended.
* PHP: PlanAway is built using the PHP programming language, and therefore, a PHP runtime environment is required. The recommended version is PHP 7.3 or higher.
* MySQL: PlanAway requires a database to store scheduling data. MySQL is recommended.
* JavaScript libraries: Various JavaScript libraries such as jQuery, Moment.js, and FullCalendar.js can be used for advanced functionality.
* Virtualization Software
* Virtualization software can be used to test PlanAway in different environments and configurations. The following virtualization software is recommended:
* VirtualBox: VirtualBox can be used to create virtual machines running different operating systems and configurations.
* Docker: Docker can be used to create containers running different software versions and configurations.
* In conclusion, PlanAway requires various software components to function correctly. By ensuring that the required software is installed and configured correctly, we can build a reliable and efficient scheduling system.
* (ii) Hardware Requirements--- Hardware requirements for PlanAway depend on the number of users and the size of the data to be stored. The following are the recommended hardware requirements for running PlanAway:
* Processor: PlanAway is a web-based application, and therefore, a modern processor is recommended for optimal performance. A processor with at least two cores and a clock speed of 2.0 GHz or higher is recommended.
* Graphics card: A dedicated graphics card is not required for running PlanAway, as it is a web-based application that runs on a browser.
* Disk capacity: The amount of disk space required depends on the size of the data to be stored. As a general rule of thumb, a minimum of 10 GB of disk space is recommended.
* RAM capacity: PlanAway is a memory-intensive application, and therefore, a minimum of 4 GB of RAM is recommended. For larger data sets or multiple concurrent users, 8 GB or more is recommended.
* In addition to the above hardware requirements, it is also recommended to use an SSD for improved performance and faster data access. The network infrastructure should also be capable of handling multiple concurrent users and the required bandwidth for data transfer.
* It is important to note that the above hardware requirements are recommended for running PlanAway on a single machine. For larger deployments, such as running PlanAway on a server, the hardware requirements will vary depending on the number of users and the size of the data to be stored. It is recommended to consult with a hardware expert or IT professional to determine the appropriate hardware requirements for running PlanAway in a larger deployment.
  + 1. **CONSTRAINTS**

Here are the constraints of the PlanAway project:

1. Hardware Limitations: The system will require certain hardware resources to function properly, such as a computer with a minimum specification of a dual-core processor, 4GB of RAM, and a stable internet connection. Users with hardware below this specification may experience performance issues while using the system.
2. Language Requirements: The proposed system will be developed using a combination of HTML, CSS, JavaScript, and PHP. Therefore, users who do not have an understanding of these programming languages may find it challenging to customize the system.
3. Interfaces to other Applications: The system will need to integrate with other applications, such as electronic health records (EHR) and student information systems (SIS), to access and share relevant data. This integration will require an API and may require additional development time and resources.
4. Reliability Requirements: The system will be used for critical scheduling purposes such as medical appointments and class schedules. Therefore, it is essential that the system is reliable and does not experience downtime or data loss. The system will need to be thoroughly tested and have a robust backup and recovery system in place.
5. Regulatory Policies: The system will need to comply with various regulatory policies, such as HIPAA for healthcare scheduling and FERPA for educational scheduling. These policies mandate specific security and privacy requirements that must be met by the system.
6. Platform Dependencies: The proposed system will be web-based and will require a web browser to function. The system will need to be compatible with multiple web browsers, such as Google Chrome, Mozilla Firefox, and Microsoft Edge. It will also need to be compatible with different operating systems, such as Windows, MacOS, and Linux.
7. Time and Resource Constraints: The proposed system will require significant time and resources to develop and implement. The project team will need to work within the given timeline and budget constraints to ensure that the system is delivered on time and within budget.
8. Scalability: The system needs to be scalable to accommodate growth in the number of users and data volume. This will require the use of a scalable database system and cloud-based hosting.
9. Security: The system will need to be secure to protect user data and prevent unauthorized access. This will require the implementation of various security measures, such as encryption, access controls, and secure login authentication.
10. Usability: The system will need to be easy to use and intuitive for users to understand. This will require the use of appropriate design and user experience (UX) principles, such as clear navigation and labeling, and effective use of color and typography.

In summary, the PlanAway project will need to consider various constraints, such as hardware limitations, language requirements, interfaces to other applications, reliability requirements, regulatory policies, platform dependencies, and time and resource constraints. Overcoming these constraints will require careful planning, design, and implementation to ensure that the system meets the needs of users while remaining secure, reliable, and scalable.

* + 1. **FUNCTIONAL REQUIREMENTS**

Functional requirements are specifications of what the system should do or what services it should provide. In the context of the PlanAway unified scheduling system, the following functional requirements have been identified:

1. User authentication: The system should have a secure and reliable user authentication mechanism that ensures only authorized users have access to the system. This requirement is crucial to ensure that sensitive information is protected from unauthorized access.
2. Schedule creation and management: The system should allow users to create and manage schedules for various domains, including hospital management scheduling and university scheduling. Users should be able to specify details such as dates, times, resources, and personnel involved in each scheduled event.
3. Automated scheduling: The system should be capable of generating schedules automatically based on predefined constraints, preferences, and rules. For instance, the system should be able to automatically generate exam schedules based on course requirements and student availability.
4. Resource allocation: The system should have a mechanism for allocating resources such as classrooms, staff, and equipment for each scheduled event. The allocation mechanism should be flexible enough to handle multiple domains and ensure optimal use of resources.
5. Conflict resolution: The system should be able to detect and resolve conflicts that arise during the scheduling process. For example, the system should be able to resolve conflicts between staff schedules or resource availability.
6. Notification and communication: The system should have a notification mechanism that alerts users of schedule changes, updates, or conflicts. The system should also provide a platform for communication between users involved in the scheduling process.
7. Reporting and analytics: The system should be capable of generating reports and analytics to help users make informed decisions. For example, the system should be able to provide statistics on staff utilization or resource allocation.
8. Customization: The system should be customizable to suit the specific needs of different users. Users should be able to configure the system to incorporate new domains, rules, and preferences.
9. Integration: The system should be able to integrate with other applications or systems to enable seamless data exchange and sharing. For example, the system should be able to integrate with hospital management systems to exchange patient information or staff schedules.
10. Security: The system should be designed with security in mind, with mechanisms in place to protect sensitive data, prevent unauthorized access, and ensure data privacy.

In summary, the functional requirements of the PlanAway unified scheduling system are designed to enable effective scheduling, resource allocation, conflict resolution, and reporting. The system is designed to be flexible, customizable, and secure, with the ability to integrate with other systems as required.

**2.6.5 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements refer to the quality attributes of a software system that specify how well it performs certain functions or how it should behave in certain conditions. These requirements describe the system's performance, reliability, security, usability, scalability, maintainability, and other characteristics that affect its overall quality.

For the proposed PlanAway system, the non-functional requirements are as follows:

1. Performance: The system should be able to handle a large number of users simultaneously without any performance degradation. It should have fast response times and minimal downtime.
2. Reliability: The system should be reliable and fault-tolerant. It should be able to recover from failures quickly and resume normal operations without any loss of data or functionality.
3. Security: The system should be secure and protect sensitive data from unauthorized access, modification, or theft. It should implement secure authentication and authorization mechanisms, data encryption, and other security measures to ensure data privacy and integrity.
4. Usability: The system should be easy to use and intuitive, with a user-friendly interface and clear instructions. It should have consistent navigation and user-friendly design, with clear feedback and error messages.
5. Scalability: The system should be able to handle increasing numbers of users, data, and transactions as the user base grows. It should be scalable and flexible, with the ability to add or remove features, modules, and resources as needed.
6. Maintainability: The system should be easy to maintain and upgrade, with modular design and clear documentation. It should support version control, code review, and other best practices to ensure code quality and maintainability.
7. Compatibility: The system should be compatible with different operating systems, web browsers, and hardware configurations. It should be tested and verified for compatibility with popular platforms and devices.
8. Accessibility: The system should be accessible to users with disabilities, with support for assistive technologies such as screen readers and keyboard navigation. It should comply with accessibility standards and guidelines to ensure equal access for all users.
9. Performance Efficiency: The system should make efficient use of computing resources, such as memory, processing power, and network bandwidth. It should be optimized for performance and efficiency, with minimal resource usage and maximum throughput.
10. Interoperability: The system should be interoperable with other systems and applications, with the ability to exchange data and communicate with external systems. It should support standard data formats and protocols to ensure interoperability with other systems.

In conclusion, these non-functional requirements play an important role in ensuring the quality and usability of the PlanAway system. These requirements should be carefully considered and incorporated into the design and development of the system to ensure that it meets the needs and expectations of its users.

**2.7 BLOCK DIAGRAM**

* Draw a diagram of the system under study, illustrating the major components and functions.
* Now that you have understood the problem domain and the requirements, produce a model of the system, reflecting the operations/functions that can be performed on/by the system and the allowable sequence of these operations.
* Conceptual Models can include Context Diagram / Class Diagrams / System Flow Charts

**Functional Requirements**

**Table 2.1 Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement | Description |
| M1\_FR1 | Access mode | Physician, Nurse and Pharmacy personnel shall able to access the system with dedicated privilege sets. Each user of the system should have his/her own set of functions and dedicated interface. |
| M1\_FR2 | Dashboard display | This should display the current order details and any updates pertaining to the patient’s medications prescriptions. |
| M1\_FR3 | Patient Health record portability | Patient health record should be transferrable across different clinical department electronically. |
| M1\_FR4 | Medication | Pharmacist should be able to place the medication orders for a patient. |
| M2\_FR5 | Med Management | Pharmacist should be able to manage the patient’s medication orders such as Validate, Revise, Discontinue etc. |
| M2\_FR6 | Search | Patient records should be able to search. |
| M3\_FR7 | Orders | Worklist Should display the current medication orders that are to be validated or functions to be performed on. |

**Non-Functional Requirements**

**Table 2.2 Non-Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement | Description |
| NF\_R1 | Performance | As it’s a Desktop application, the network, hardware and other related infrastructure plays a vital role in determining the application performance.  • Data compression approach has been applied to reduce the network burden.  • Number of window navigations and clicks are reduced to the minimum.  • Basic and simple color and graphics settings are implemented to help the  performance.  • Disabling browsing the server. |
| NF\_R2 | Safety/Security | Performing frequent backup can reduces the data loss due to sudden server or the system  crash. Being a healthcare workflow system, its primary character should be security,  thus providing secure environment for the  patient flow process. This system allows the Physician ordering web application to only be accessed over the secure Cerner network  and with his/her Cerner access credentials of the system. |
| NF\_R3 | Quality Requirements | • **Usability**  As the users of the system are from various clinical departments and with different  level of functionalities should require better understandability of the system. The interface for each type of user kept very  simple and complete for ease of its user. Manuals, demos or the documents made available, simple, clear and definite set of  interfaces makes the context easy to understand and use.  • **Reliability** This system uses atomicity features where  each task flow is performed to complete a process successful, failure of single task halts the transaction and the process fails. This avoids occurrence of incomplete order processing. Data backup ensures the  availability of data to the system users all the time.  • **Availability** This web application handles multiple  service requests and the server is up all the time, which is made sure by the robust algorithms and server design architecture. |

**3. SYSTEM DESIGN**

**3.1 SYSTEM ARCHITECTURE**

The system architecture of the PlanAway project is designed to provide an efficient and robust solution for unified scheduling. It is built using a layered architecture approach, which provides a clear separation of concerns and ensures that the system is modular and scalable.

The architecture is composed of four main layers:

1. Presentation Layer: This layer is responsible for rendering the user interface and handling user input. It is built using HTML, CSS, and JavaScript, and runs on the client-side. The presentation layer is designed to be flexible, allowing for customization based on user preferences.
2. Application Layer: The application layer acts as an intermediary between the presentation layer and the business logic layer. It is responsible for processing user input, performing validations, and handling errors. The application layer is implemented using PHP, which is a server-side scripting language that is widely used for web development.
3. Business Logic Layer: This layer contains the core logic of the system. It is responsible for handling all the business rules and requirements of the project. The business logic layer is implemented using PHP, and it interacts with the database layer to retrieve and manipulate data.
4. Database Layer: The database layer is responsible for storing and retrieving data. It is built using MySQL, which is an open-source relational database management system. The database layer is designed to be scalable, ensuring that the system can handle large amounts of data.

The architecture also includes several interfaces to other applications. For instance, the system can interface with hospital information systems (HIS) and university management systems (UMS). This allows for seamless integration of data between different systems, making it easier for users to manage their schedules.

To ensure reliability, the system architecture includes several features, such as data backup and recovery mechanisms. This ensures that data is not lost in the event of system failure or downtime. The system also includes several security features, such as user authentication and authorization, data encryption, and secure transmission of data over the network.

The architecture is designed to be platform-independent, allowing it to run on different operating systems, such as Windows, macOS, and Linux. However, it is important to note that the system has some hardware limitations. For instance, it requires a stable internet connection and a web browser to function properly.

In summary, the system architecture of the PlanAway project is designed to provide an efficient, robust, and scalable solution for unified scheduling. The layered architecture approach ensures that the system is modular and easy to maintain, while the interfaces to other applications allow for seamless integration of data. The inclusion of several security and reliability features ensures that the system is secure and reliable.

3**.2 MODULE DESIGN**

The module design of PlanAway includes various components that are designed to cater to the specific needs of the hospital and university domains. Each module is designed to provide maximum functionality and user-friendliness. Below are the details of each module:

1. Doctor Patient Scheduling: This module allows doctors to manage their patient appointments efficiently. The module includes features such as setting up patient appointments, rescheduling appointments, and canceling appointments. The module also provides reminders to doctors and patients before appointments to ensure that appointments are not missed.
2. Clinic Scheduling: This module allows the hospital to manage clinic schedules efficiently. The module includes features such as scheduling appointments for different clinics, rescheduling appointments, and canceling appointments. The module also provides notifications to patients and doctors regarding the schedule.
3. Staff Shift Scheduling: This module allows the hospital to manage the shifts of their staff members efficiently. The module includes features such as scheduling staff shifts, changing shifts, and canceling shifts. The module also provides notifications to staff members regarding their shifts.
4. Automated Timetable Generator: This module allows the university to generate automated timetables for various departments and courses. The module includes features such as selecting courses, selecting teachers, and scheduling classes. The module also provides notifications to teachers and students regarding their schedules.
5. Classroom Scheduling: This module allows the university to manage classroom schedules efficiently. The module includes features such as scheduling classes for different courses, rescheduling classes, and canceling classes. The module also provides notifications to teachers and students regarding their schedules.
6. Exam Scheduling: This module allows the university to schedule exams for various courses and departments. The module includes features such as selecting courses, selecting teachers, and scheduling exams. The module also provides notifications to teachers and students regarding their exam schedules.

Apart from these modules, the PlanAway system includes various other features that enhance its functionality and usability. These features include:

1. Customizable Dashboard: The dashboard can be customized according to the user's preferences. The user can select which modules they want to display on their dashboard.
2. Role-Based Access Control: The system provides different levels of access to different users based on their roles. For example, doctors have access to patient scheduling modules, while staff members have access to staff shift scheduling modules.
3. Reporting and Analytics: The system provides various reports and analytics to help hospital and university administrators make informed decisions. Reports such as patient and staff attendance, scheduling data, and utilization data help administrators understand the efficiency of the system.
4. Mobile App: The system includes a mobile app that allows doctors, staff members, teachers, and students to access the system from their mobile devices. The app includes all the features of the web application, making it easier for users to manage their schedules on-the-go.

The modules of PlanAway are designed to work together seamlessly to provide maximum functionality and efficiency. The system is flexible and can be customized according to the needs of the hospital and university. The features provided by the system help hospital and university administrators manage their schedules more efficiently, saving time and resources.

In this section, you should explain all the modules of your project along with their functionalities.

**3.3 DATA FLOWDIAGRAM**

Diagrammatic representation of the flow of data with 3 levels of information

**3.4 ER DIAGRAM**

Diagrammatic representation of entities and relationships of the project

**3.5 DATABASE DESIGN**

In this section, you should explain how you have organized, arranged, managed and manipulated the data pertaining to your project. This section should explain the mapping of entities & relationships to tables.

**3.5.1 Table Design:** Define the structure of each table and its relationship with other tables in the database.

Table 3.1 Table Name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Attribute name | Data Type | Description | Constraint |
|  |  |  |  |  |

**3.5.2 Data Integrity and Constraints:**

Data integrity is a critical aspect of any software application, and the PlanAway project is no exception. The system will be handling sensitive information related to patients, doctors, staff, students, courses, classrooms, and exams, among others. It is essential to ensure that the data is accurate, complete, and consistent throughout the system.

To achieve data integrity, the PlanAway system will implement various constraints to ensure that the data entered into the system meets specific requirements. These constraints will be enforced through validation rules and data entry controls.

1. Data Validation Rules: The PlanAway system will implement data validation rules to ensure that the data entered into the system is valid and accurate. These rules will ensure that the data entered meets specific requirements, such as data type, format, and range. For example, when entering patient data, the system will ensure that the data entered is in the correct format, including the patient's name, address, date of birth, and medical record number.
2. Data Entry Controls: Data entry controls are an essential aspect of data integrity in the PlanAway system. These controls will ensure that the data entered is complete and accurate. The system will implement various controls, such as mandatory fields, drop-down lists, and default values, to ensure that the data entered is accurate and complete.
3. Access Controls: Access controls are an essential aspect of data integrity in the PlanAway system. The system will implement various access controls to ensure that only authorized personnel can access the data. These controls will include user authentication, authorization, and access permissions.
4. Referential Integrity: Referential integrity is a critical aspect of data integrity in the PlanAway system. It ensures that the relationships between the data entities are maintained correctly. The system will implement referential integrity constraints to ensure that the data relationships are consistent throughout the system.
5. Concurrency Control: Concurrency control is an essential aspect of data integrity in the PlanAway system. It ensures that multiple users accessing the same data do not interfere with each other's changes. The system will implement concurrency control mechanisms to ensure that only one user can access and modify the data at a time.

In addition to the above constraints, the PlanAway system will also implement various security measures to ensure data integrity. These measures will include encryption of sensitive data, regular backups, disaster recovery plans, and access audits.

The hardware limitations of the PlanAway system include the need for sufficient processing power, memory, and storage to handle the large volumes of data. The system will require modern hardware with adequate processing power, memory, and storage to ensure smooth operation.

The language requirements of the PlanAway system include the need for expertise in HTML, CSS, JavaScript, and PHP. These programming languages are essential for the development of the front-end and back-end components of the system.

The interfaces to other applications are also a critical aspect of the PlanAway system. The system will need to integrate with various third-party applications, such as electronic medical record systems, student information systems, and scheduling systems, among others. The system will need to be designed to support various interface standards, such as REST APIs, SOAP APIs, and XML interfaces.

Reliability requirements for the PlanAway system include the need for high availability and fault tolerance. The system will need to be designed to operate 24/7, and any downtime must be kept to a minimum. The system will need to be fault-tolerant, with backup systems and disaster recovery plans in place.

Regulatory policies will also be a constraint on the PlanAway system. The system will need to comply with various regulatory policies, such as HIPAA for patient data, FERPA for student data, and GDPR for personal data. The system will need to implement various security measures to ensure compliance with these policies.

**3.5.3 Data Dictionary**

Table 3.2 Table Name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Attribute name | Min Value | Max Value | Default Value |
|  |  |  |  |  |

**3.6 INTERFACE AND PROCEDURAL DESIGN**

**3.6.1 USER INTERFACE DESIGN:**

* Define user, the task being performed, the environment in which task is performed and how you intend to map those requirements to develop a “User Interface”.
* Illustrate the interface with the help of some rough pictorial views of the User Interface which helps the programmer during implementation.

(5 to 10 UI screens)

**3.6.2 PROCEDURAL DESIGN:**

* Define the core algorithm used in designing the project

**//API or functions used or settings needed**

**3.7 REPORTS DESIGN:**

Define

(i) The structure of the reports that will be generated by the system

(ii) Inputs given to generate the report (eg. Time period, Name of product etc.)

(iii) Output fields in the generated report etc.

(3 report screens to be included)

**Chapter 4: Implementation**

Begin this chapter with a summary of the contents of this chapter (Maximum two lines).

**4.1 Coding Standards:** Define the coding standards used for writing the code of your project.

**LOGIN PAGE CSS :**

body {

font-family: "Montserrat", sans-serif;

background: #d5f5f7; }

.container {

max-width: 900px; }

a {

display: inline-block;

text-decoration: none; }

input {

outline: none !important; }

h1 {

text-align: center;

text-transform: uppercase;

margin-bottom: 40px;

font-weight: 700; }

section#formHolder {

padding: 50px 0; }

.brand {

padding: 20px;

background: url(https://goo.gl/A0ynht);

background-size: cover;

background-position: center center;

color: #fff;

min-height: 540px;

position: relative;

box-shadow: 3px 3px 10px rgba(0, 0, 0, 0.3);

transition: all 0.6s cubic-bezier(1, -0.375, 0.285, 0.995);

z-index: 9999; }

.brand.active {

width: 100%; }

.brand::before {

content: '';

display: block;

width: 100%;

height: 100%;

position: absolute;

top: 0;

left: 0;

background: rgba(0, 0, 0, 0.85);

z-index: -1; }

.brand a.logo {

color: #f95959;

font-size: 20px;

font-weight: 700;

text-decoration: none;

line-height: 1em; }

.brand a.logo span {

font-size: 30px;

color: #fff;

transform: translateX(-5px);

display: inline-block; }

.brand .heading {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

text-align: center;

transition: all 0.6s; }

.brand .heading.active {

top: 100px;

left: 100px;

transform: translate(0); }

.brand .heading h2 {

font-size: 70px;

font-weight: 700;

text-transform: uppercase;

margin-bottom: 0; }

.brand .heading p {

font-size: 15px;

font-weight: 300;

text-transform: uppercase;

letter-spacing: 2px;

white-space: 4px;

font-family: "Raleway", sans-serif; }

.brand .success-msg {

width: 100%;

text-align: center;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

margin-top: 60px; }

.brand .success-msg p {

font-size: 25px;

font-weight: 400;

font-family: "Raleway", sans-serif; }

.brand .success-msg a {

font-size: 12px;

text-transform: uppercase;

padding: 8px 30px;

background: #f95959;

text-decoration: none;

color: #fff;

border-radius: 30px; }

.brand .success-msg p, .brand .success-msg a {

transition: all 0.9s;

transform: translateY(20px);

opacity: 0; }

.brand .success-msg p.active, .brand .success-msg a.active {

transform: translateY(0);

opacity: 1; }

.form {

position: relative; }

.form .form-peice {

background: #fff;

min-height: 480px;

margin-top: 30px;

box-shadow: 3px 3px 10px rgba(0, 0, 0, 0.2);

color: #bbbbbb;

padding: 30px 0 60px;

transition: all 0.9s cubic-bezier(1, -0.375, 0.285, 0.995);

position: absolute;

top: 0;

left: -30%;

width: 130%;

overflow: hidden; }

.form .form-peice.switched {

transform: translateX(-100%);

width: 100%;

left: 0; }

.form form {

padding: 0 40px;

margin: 0;

width: 70%;

position: absolute;

top: 50%;

left: 60%;

transform: translate(-50%, -50%); }

.form form .form-group {

margin-bottom: 5px;

position: relative; }

.form form .form-group.hasError input {

border-color: #f95959 !important; }

.form form .form-group.hasError label {

color: #f95959 !important; }

.form form label {

font-size: 12px;

font-weight: 400;

text-transform: uppercase;

font-family: "Montserrat", sans-serif;

transform: translateY(40px);

transition: all 0.4s;

cursor: text;

z-index: -1; }

.form form label.active {

transform: translateY(10px);

font-size: 10px; }

.form form label.fontSwitch {

font-family: "Raleway", sans-serif !important;

font-weight: 600; }

.form form input:not([type=submit]) {

background: none;

outline: none;

border: none;

display: block;

padding: 10px 0;

width: 100%;

border-bottom: 1px solid #eee;

color: #444;

font-size: 15px;

font-family: "Montserrat", sans-serif;

z-index: 1; }

.form form input:not([type=submit]).hasError {

border-color: #f95959; }

.form form span.error {

color: #f95959;

font-family: "Montserrat", sans-serif;

font-size: 12px;

position: absolute;

bottom: -20px;

right: 0;

display: none; }

.form form input[type=password] {

color: #f95959; }

.form form .CTA {

margin-top: 30px; }

.form form .CTA input {

font-size: 12px;

text-transform: uppercase;

padding: 5px 30px;

background: #f95959;

color: #fff;

border-radius: 30px;

margin-right: 20px;

border: none;

font-family: "Montserrat", sans-serif; }

.form form .CTA a.switch {

font-size: 13px;

font-weight: 400;

font-family: "Montserrat", sans-serif;

color: #bbbbbb;

text-decoration: underline;

transition: all 0.3s; }

.form form .CTA a.switch:hover {

color: #f95959; }

footer {

text-align: center; }

footer p {

color: #777; }

footer p a, footer p a:focus {

color: #b8b09f;

transition: all .3s;

text-decoration: none !important; }

footer p a:hover, footer p a:focus:hover {

color: #f95959; }

@media (max-width: 768px) {

.container {

overflow: hidden; }

section#formHolder {

padding: 0; }

section#formHolder div.brand {

min-height: 200px !important; }

section#formHolder div.brand.active {

min-height: 100vh !important; }

section#formHolder div.brand .heading.active {

top: 200px;

left: 50%;

transform: translate(-50%, -50%); }

section#formHolder div.brand .success-msg p {

font-size: 16px; }

section#formHolder div.brand .success-msg a {

padding: 5px 30px;

font-size: 10px; }

section#formHolder .form {

width: 80vw;

min-height: 500px;

margin-left: 10vw; }

section#formHolder .form .form-peice {

margin: 0;

top: 0;

left: 0;

width: 100% !important;

transition: all .5s ease-in-out; }

section#formHolder .form .form-peice.switched {

transform: translateY(-100%);

width: 100%;

left: 0; }

section#formHolder .form .form-peice > form {

width: 100% !important;

padding: 60px;

left: 50%; } }

@media (max-width: 480px) {

section#formHolder .form {

width: 100vw;

margin-left: 0; }

h2 {

font-size: 50px !important; } }

**DASHBOARD CSS :**   
**A)**

.modal {

display: none; /\* Hidden by default \*/

position: fixed; /\* Stay in place \*/

z-index: 1; /\* Sit on top \*/

left: 0;

top: 0;

opacity: 1;

width: 100%; /\* Full width \*/

height: 100%; /\* Full height \*/

overflow: auto; /\* Enable scroll if needed \*/

background-color: rgb(0, 0, 0); /\* Fallback color \*/

background-color: rgba(0, 0, 0, 0.4); /\* Black w/ opacity \*/

-webkit-animation-name: fadeIn; /\* Fade in the background \*/

-webkit-animation-duration: 0.4s;

animation-name: fadeIn;

animation-duration: 0.4s

}

/\* Modal Content \*/

.modal-content {

position: fixed;

top: 25%;

left: 35%;

background-color: #fff;

width: 30%;

-webkit-animation-name: slideIn;

-webkit-animation-duration: 0.4s;

animation-name: slideIn;

animation-duration: 0.4s

}

/\* The Close Button \*/

.close {

color: white;

float: right;

font-size: 28px;

font-weight: bold;

}

.close:hover,

.close:focus {

color: #000;

text-decoration: none;

cursor: pointer;

}

.modal-header {

padding: 2px 16px;

background-color: #000000;

color: white;

}

.modal-body {

padding: 2px 16px;

}

.modal-footer {

padding: 2px 16px;

background-color: #FFFFFF;

color: white;

}

/\* Add Animation \*/

@-webkit-keyframes slideIn {

from {

top: 0;

opacity: 0

}

to {

top: 25%;

opacity: 1

}

}

@keyframes slideIn {

from {

top: 0;

opacity: 0

}

to {

top: 25%;

opacity: 1

}

}

@-webkit-keyframes fadeIn {

from {

opacity: 0

}

to {

opacity: 1

}

}

@keyframes fadeIn {

from {

opacity: 0

}

to {

opacity: 1

}

}

**B)**

/\* The side navigation menu \*/

body {

background-color: #f2f2f2;

}

.main {

height: 85%;

width: 73%;

margin: 5px;

padding: 30px;

position:fixed;

margin-left: 200px;

background-color: rgba(255, 255, 255, 0.8);

border-radius: 20px;

}

**C)**

.timetablecenter{

display: flex;

justify-content: center

}

.timetable{

border: 2px solid;

width: 70%;

}

.timetable th{

border: 2px solid;

}

timetable td{

align-items: center;

border: 2px solid;

}

**COMPONENTS CSS :**

A)

<div id="leftsidenavbar" class="sidenav leftsidenav">

<a href="/dashboard/teacher/addteacher.php">Teacher</a>

<a href="#">Class</a>

<a href="#">School</a>

<a href="#">Contact</a>

</div>

<style>

.leftsidenav{

padding-right: 25px;

left: -150px;

padding-left: 60px;

text-align: right;

}

.sidenav {

margin-top: 5%;

height: 80%;

width: 250px;

position: fixed; /\* Stay in place \*/

z-index: 1; /\* Stay on top \*/

background-color: rgba(255, 255, 255, .20);

backdrop-filter: blur(5px);

border: 2px solid #4a4949;

overflow-x: hidden;

border-radius: 25px;

}

/\* The navigation menu links \*/

.sidenav a {

padding: 8px 8px 8px 32px;

text-decoration: none;

font-size: 25px;

color: #818181;

display: block;

transition: 0.3s;

}

/\* When you mouse over the navigation links, change their color \*/

.sidenav a:hover {

color: #f1f1f1;

}

/\* Position and style the close button (top right corner) \*/

.sidenav .closebtn {

position: absolute;

top: 0;

right: 25px;

font-size: 36px;

margin-left: 50px;

}

</style>

**B)**

<div id="mySidenav" class="sidenav rightsidenav">

<a href="#">Subject</a>

<a href="#">Services</a>

<a href="#">Clients</a>

</div>

<style>

.rightsidenav {

right: -150px;

padding-right: 60px;

}

</style>

**ADD TEACHER :**

<?php

include\_once("../../database/auth.php");

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">

<link rel="stylesheet" href="../css/style.css">

<link rel="stylesheet" href="../css/addteacher.css">

</head>

<body>

<?php

include\_once("../components/leftside.php");

include\_once("../components/rightside.php");

?>

<div class="main">

<div align="center" style="margin-top:20px">

<button id="teachermanual" class="btn btn-success btn-lg">ADD TEACHER</button>

</div>

<div>

<br>

<style>

table {

margin-top: 10px;

font-family: arial, sans-serif;

border-collapse: collapse;

margin-left: 30px;

width: 90%;

}

td,

th {

border: 1px solid #dddddd;

text-align: left;

padding: 8px;

}

tr:nth-child(even) {

background-color: #dddddd;

}

</style>

<script>

function deleteHandlers() {

var table = document.getElementById("teacherstable");

var rows = table.getElementsByTagName("tr");

for (i = 0; i < rows.length; i++) {

var currentRow = table.rows[i];

//var b = currentRow.getElementsByTagName("td")[0];

var createDeleteHandler =

function(row) {

return function() {

var cell = row.getElementsByTagName("td")[0];

var id = cell.innerHTML;

var x;

if (confirm("Are You Sure?") == true) {

window.location.href = "deleteteacher.php?name=" + id;

}

};

};

currentRow.cells[6].onclick = createDeleteHandler(currentRow);

}

}

</script>

<caption><strong>Teacher's Information </strong></caption>

<table id=teacherstable style="margin-left: 80px">

<tr>

<th width="130">Faculty No</th>

<th width=290>Name</th>

<th width="190">Department</th>

<th width="190">Designation</th>

<th width="290">Email ID</th>

<th width="40">Action</th>

</tr>

<tbody>

<?php

include 'database/auth.php';

$q = mysqli\_query(

$conn,

"SELECT \* FROM teacher\_list ORDER BY srno ASC"

);

while ($row = mysqli\_fetch\_assoc($q)) {

echo "<tr><td>{$row['srno']}</td>

<td>{$row['name']}</td>

<td>{$row['department']}</td>

<td>{$row['designation']}</td>

<td>{$row['email']}</td>

<td>

<button>Delete</button></td>

</tr>\n";

}

echo "<script>deleteHandlers();</script>";

?>

</tbody>

</table>

</div>

</div>

**DATABASE :**

**ADD TEACHER**

<?php

require("./auth.php");

if (isset($\_POST['TF']) && isset($\_POST['TN']) && isset($\_POST['TDP']) && isset($\_POST['TD']) && isset($\_POST['TE'])) {

$facno = $\_POST['TF'];

$name = $\_POST['TN'];

$department = $\_POST['TDP'];

$designation = $\_POST['TD'];

$email = $\_POST['TE'];

// $message = "nTry again.";

// echo "<script type='text/javascript'>alert('$message');</script>";

} else {

$message = "dead.";

echo "<script type='text/javascript'>alert('$message');</script>";

}

$sql = "INSERT INTO teacher\_list VALUES ('$facno','$name','$department','$designation','$email')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

$message = "Teacher added.\\nTry again.";

echo "<script type='text/javascript'>alert('$message');</script>";

header("Location:../addteacher.php");

} else {

$message = "Error: " . $sql . "<br>" . $conn->error."\\nTry again.";

echo "<script type='text/javascript'>alert('$message');</script>";

header("Location:../addteacher.php");

}

?>

<!-- Add teacher modal popup -->

<div id="myModal" class="modal">

<!-- Modal content -->

<div class="modal-content" style="margin-top: -60px">

<div class="modal-header">

<span class="close">&times</span>

<h2 id="popupHead">Add Teacher</h2>

</div>

<div class="modal-body" id="EnterTeacher">

<!--Admin Login Form-->

<div style="display:none" id="addTeacherForm">

<form action="database/addteacherdatabase.php" method="post">

<div class="form-group">

<label for="TF">Faculty No</label>

<input type="text" class="form-control" id="facultyno" name="TF" placeholder="Faculty No ...">

</div>

<div class="form-group">

<label for="teachername">Teacher's Name</label>

<input type="text" class="form-control" id="teachername" name="TN" placeholder="Teacher's Name ...">

</div>

<div class="form-group">

<label for="dp">Department</label>

<input type="text" class="form-control" id="department" name="TDP" placeholder="Department ...">

</div>

<div class="form-group">

<label for="designation">Designation</label>

<select class="form-control" id="designation" name="TD">

<option selected disabled>Select</option>

<option value="HOD">Head Of the Department</option>

<option value="Professor">Professor</option>

<option value="Associate Professor">Associate Professor</option>

<option value="Assistant Professor">Assistant Professor</option>

<option value="Guest Faculty">Guest Faculty</option>

</select>

</div>

<div class="form-group">

<label for="teacheremailid">Email-ID</label>

<input type="text" class="form-control" id="teacheremailid" name="TE" placeholder="abc@xyz.com ...">

</div>

<div align="right">

<input type="submit" class="btn btn-default" name="ADD" value="ADD">

</div>

</form>

</div>

</div>

<div class="modal-footer">

</div>

</div>

</div>

<script>

// Get the modal

var modal = document.getElementById('myModal');

// Get the button that opens the modal

var addteacherBtn = document.getElementById("teachermanual");

var heading = document.getElementById("popupHead");

var facultyForm = document.getElementById("addTeacherForm");

// Get the <span> element that closes the modal

var span = document.getElementsByClassName("close")[0];

// When the user clicks the button, open the modal

addteacherBtn.onclick = function() {

modal.style.display = "block";

//heading.innerHTML = "Faculty Login";

facultyForm.style.display = "block";

//adminForm.style.display = "none";

}

// When the user clicks on <span> (x), close the modal

span.onclick = function() {

modal.style.display = "none";

//adminForm.style.display = "none";

facultyForm.style.display = "none";

}

// When the user clicks anywhere outside of the modal, close it

window.onclick = function(event) {

if (event.target == modal) {

modal.style.display = "none";

}

}

</script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>

<script src="../js/script.js"></script>

</body>

</html>

**AUTHENTICATION :**

<?php

$servername = "localhost";

$username = "planaway";

$password = "planaway";

$db = "planaway";

// Create connection

$conn = new mysqli($servername, $username, $password,$db

);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

else{

}

?>

**TIMETABLE :**

<?php

include\_once("../database/auth.php");

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link rel="stylesheet" href="css/style.css">

<link rel="stylesheet" href="css/timetable.css">

</head>

<body>

<?php

include\_once("./components/leftside.php");

include\_once("./components/rightside.php");

?>

<div class="main">

<?php

$sem\_class = "6BCANCR";

?>

<h3 style="text-align:center;">

Class Name :

<?php

echo $sem\_class. "\t\t";

?>

<span style="text-align:right;">

Class Teacher:

<?php

$sql = "SELECT \* FROM classes where sem='" . $sem\_class

. "'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

// output data of each row

$row = $result->fetch\_assoc();

echo $row['class\_teacher'] . "<br>";

}

?>

</span>

</h3>

<div class="timetablecenter">

<table class="timetable">

<tr>

<th></th>

<th>1</th>

<th>2</th>

<th>3</th>

<th>4</th>

<th rowspan="7">Lunch</th>

<th>5</th>

<th>6</th>

</tr>

<?php

$days = array(

0 => 'monday',

1 => 'tuesday',

2 => 'wednesday',

3 => 'thursday',

4 => 'friday',

5 => 'saturday',

);

for ($i = 0; $i < 6; $i++) {

?> <tr>

<?php

$sql = "SELECT \* FROM timetable\_" . $days[$i] . " where class\_name='" . $sem\_class . "'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

// output data of each row

$row = $result->fetch\_assoc();

} else {

echo "0 results";

}

?>

<th><?php echo strtoupper($days[$i]); ?></th>

<?php

for ($j = 1; $j < 7; $j++) {

echo "<td>";

$sql1 = "SELECT \* FROM timetable\_mapping where subject\_code='" . $row['period\_' . $j . ''] . "'";

$result1 = $conn->query($sql1);

if ($result1->num\_rows > 0) {

// output data of each row

$row1 = $result1->fetch\_assoc();

echo $row1['teacher\_name'] . "<br>";

}

$sql2 = "SELECT \* FROM timetable\_subject where subject\_code='" . $row['period\_' . $j . ''] . "'";

$result2 = $conn->query($sql2);

if ($result2->num\_rows > 0) {

// output data of each row

$row2 = $result2->fetch\_assoc();

echo $row2['subject\_name'] . "<br>";

}

echo $row['period\_' . $j . ''] . "</td>";

}

?>

</tr>

<?php

}

?>

</table>

</div>

</div>

<script src="js/script.js"></script>

</body>

</html>

**INDEX.PHP**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Document</title>

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"

rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"

crossorigin="anonymous"

/>

<link rel="stylesheet" href="/css/style.css" />

</head>

<body>

<div class="container">

<section id="formHolder">

<div class="row">

<!-- Brand Box -->

<div class="col-sm-6 brand">

<a href="#" class="logo">Nothing <span>.</span></a>

<div class="heading">

<img src="./image/logo.png" alt="logo" width="100%" srcset="">

</div>

<div class="success-msg">

<p>Great! You are one of our members now</p>

<a href="#" class="profile">Your Profile</a>

</div>

</div>

<!-- Form Box -->

<div class="col-sm-6 form">

<!-- Login Form -->

<div class="login form-peice switched">

<form class="login-form" action="#" method="post">

<div class="form-group">

<label for="loginemail">Email Adderss</label>

<input

type="email"

name="loginemail"

id="loginemail"

required

/>

</div>

<div class="form-group">

<label for="loginPassword">Password</label>

<input

type="password"

name="loginPassword"

id="loginPassword"

required

/>

</div>

<div class="CTA">

<input type="submit" value="Login" />

<a href="#" class="switch">I'm New</a>

</div>

</form>

</div>

<!-- End Login Form -->

<!-- Signup Form -->

<div class="signup form-peice">

<form class="signup-form" action="./database/createuser.php" method="post">

<div class="form-group">

<label for="name">Full Name</label>

<input type="text" name="username" id="name" class="name" />

<span class="error"></span>

</div>

<div class="form-group">

<label for="email">Email Adderss</label>

<input

type="email"

name="email"

id="email"

class="email"

/>

<span class="error"></span>

</div>

<div class="form-group">

<label for="phone"

>Phone Number - <small>Optional</small></label

>

<input type="text" name="phone" id="phone" />

</div>

<div class="form-group">

<label for="password">Password</label>

<input

type="password"

name="password"

id="password"

class="pass"

/>

<span class="error"></span>

</div>

<div class="form-group">

<label for="passwordCon">Confirm Password</label>

<input

type="password"

name="passwordCon"

id="passwordCon"

class="passConfirm"

/>

<span class="error"></span>

</div>

<div class="CTA">

<input type="submit" value="Signup Now" id="submit" />

<a href="#" class="switch">I have an account</a>

</div>

</form>

</div>

<!-- End Signup Form -->

</div>

</div>

</section>

</div>

<script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"

integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN"

crossorigin="anonymous"

></script>

<script

src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.3/jquery.min.js"

integrity="sha512-STof4xm1wgkfm7heWqFJVn58Hm3EtS31XFaagaa8VMReCXAkQnJZ+jEy8PCC/iT18dFy95WcExNHFTqLyp72eQ=="

crossorigin="anonymous"

referrerpolicy="no-referrer"

></script>

<script src="/js/script.js"></script>

</body>

</html>

**4.2 Coding Details:**  Include only the important portions of code (applets code, forms code etc.) The lines of code should be commented appropriately for explaining the purpose of a piece of code. Screen shot of the output of the program code can be included to demonstrate the functionality of the code.

**4.3 Screen Shots:** Input Screen shots & Output Screen shots.

**\*\*\*Maximum of 15 pages of code**

**\*\*\*Maximum of 12 output screen shots**

## 4.1 CODING STANDARDS

Implementation is the process of completing the system and turning it over to the user. Implementation is the part of the process where software engineers actually program the code for the project. In the implementation phase, the team builds the components either from scratch or by composition. Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility. For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline. The architecture document should give guidance. Sometimes, this guidance is found in the requirement document. The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself.

Code conventions are important to programmers for a number of reasons:

* 80% of the lifetime cost of a piece of software goes to maintenance. Hardly any software is maintained for its whole life by the original author.
* Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
* If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.
* Indenting and Line Length - Use an indent of 4 spaces and don't use any tab because different computers use different setting for tab. It is recommended to keep lines at approximately 75-85 characters long for better code readability.
* Control Structures - These include if, for, while, switch, etc. Control statements should have one space between the control keyword and opening parenthesis, to distinguish them from function calls. You are strongly encouraged to always use curly braces even in situations where they are technically optional.

if ((condition1) || (condition2)) {

if ((condition1) || (condition2)) {

action1;

} elseif ((condition3) && (condition4)) {

action2;

} else {

default action;

}

* Function Calls - Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon. Here's an example:

$var = foo($bar, $baz, $quux);

Function Definitions - Function declarations follow the "BSD/Allman style":

function fooFunction($arg1, $arg2 = '')

{

if (condition) {

statement;

}

return $val;

}

* Comments - C style comments (/\* \*/) and standard C++ comments (//) are both fine. Use of Perl/shell style comments (#) is discouraged.
* Variable Names -
  + Use all lower case letters
  + Use '\_' as the word separator.
  + Global variables should be prepended with a 'g'.
  + Global constants should be all caps with '\_' separators.
  + Static variables may be prepended with’s’.
* Make Functions Reentrant - Functions should not keep static variables that prevent a function from being reentrant.
* Alignment of Declaration Blocks - Block of declarations should be aligned.
* One Statement Per Line - There should be only one statement per line unless the statements are very closely related.
* Short Methods or Functions - Methods should limit themselves to a single page of code.

**Chapter 5: Testing**

Begin this chapter with a summary of the contents of this chapter (Maximum two lines).

* 1. **Testing Approaches:** Explain the approach of testing.

**Unit Testing**

**Integration Testing**

**5.2 Test Cases:** Test cases are all about “How you are going to test a requirement?” Define the test cases for various test scenarios in your project.

Explain the purpose of each test (test case description) and expected result in the below mentioned format.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sl.No | Module Name | Test Case No | Test Case Description | Expected Result |
| 1 | REGISTRATION MODULE | TC1 | Verify the user can register with different values in password and confirm password | To validate the name and Password |
| 2 | REGISTRATION MODULE | TC2 | Verify the registration form when the field is blank and submit button is clicked | To enter the required details |

* 1. **Test Reports:** Explain the test results and reports based on your test cases, which would prove that your project will produce the expected results under different conditions. Produce the test reports as a screen shot for different test cases by giving different sample inputs and showing the corresponding outputs.

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.No | Test Case No | Test Status | TEST Report |
| 1 | TC1 | Successful | Fig. 6.1 |
| 2 | TC2 | Successful | Fig 6.2 |

**Chapter 6: Conclusion**

Begin this chapter with a summary of the contents of this chapter (Maximum two lines).

* 1. **Design and Implementation Issues**

Poor design and/or implementation can cause failure or rejection of a software system.

**6.1.1 Design Issues:** Explain the challenges such as requirements volatility, design process, quality issues (e.g., scalability, performance, security, usability etc.), distributed software development, fast-changing technology etc. faced during the design of your project.

**6.1.2 Implementation Issues:**  Explain the challenges faced by your system during implementation such as operating environment, installation of the system, code conversion, change over, training etc.

* 1. **Advantages and Limitations**

**Advantages:** Explain how a problem identified with the existing system has been resolved (ii) Contribution of your work towards improving the performance of the existing system.

**Limitations:** (i) Explain the functionalities/requirements you were not able implement partially or completely (ii) limitations identified by you during the testing of your project that you were not able to modify (iii) Suggestions for improvement received during demonstration of your project in your company.

* 1. **Future Scope of the Project:** (i) Discuss the new related areas of investigation identified during the development of the project. (ii) Parts of the current work that could not be completed due to time constraints and/or problems encountered.