UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: A Framework for Web Application Development in Smart Campus Environments

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor:

Arévalo Torres Carlos Andrés

Tutor:

Iza Paredes Cristhian René, PhD.

Urcuquí, Agosto de 2024

Autoría

Yo, **Carlos Andrés Arévalo Torres**, con cédula de identidad 0706497443, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Agosto de 2024.

Firmado electrónicamente por:
CARLOS ANDRES
AREVALO TORRES

Carlos Andrés Arévalo Torres
CI: 0706497443

Autorización de publicación

Dedication

This work is dedicated to my family, whose support and love have been my guiding light. To my mother, for instilling in me the value of perseverance and to my brother for his endless encouragement. To my friends: Juan, Samantha, Maithé, Joel, Julio, Harvey, Cristhian, Ricardo, Danny, Franz, Lis, Mariu, Anthony, Emilio, Milena, Juanito, Gustavo, Jeff, Manuel, Iván, Mabelen and any who made part of this adventure into science and tech during university. Also, an exclusive dedication to my best friend Kary: thank you for everything. This achievement represents the collective dreams and sacrifices of my family, a chapter of our shared journey towards growth and excellence where I made unforgettable friends.

Carlos Andrés Arévalo Torres

Acknowledgment

I extend my sincerest gratitude to all those who have contributed to the process of this thesis. I thank my academic advisor, whose guidance have been indispensable in my research. My appreciation also goes to my teachers and peers for their encouragement along the career. Special thanks to Yachay Tech staff who allows me to apply part of my thesis work inside the university community, without IT Department and volunteer tutors for the proof-of-concept application, the pilot project couldn't be possible. Your collective support and wisdom were important to me.

Carlos Andrés Arévalo Torres

Resumen

En los últimos años, se ha observado un cambio notable en el panorama educativo, que ha evolucionado de los campus digitales tradicionales a sofisticados ecosistemas de campus inteligentes. Esta evolución subraya la necesidad crítica de aprovechar las tecnologías avanzadas para optimizar la eficiencia operativa y la prestación de servicios dentro de las instituciones académicas. A pesar de los avances realizados por Ecuador en el fortalecimiento de su infraestructura de tecnología educativa, el establecimiento de campus inteligentes aún se encuentra en su fase incipiente, lo que requiere esfuerzos concertados para alinearse con las tendencias educativas globales. Esta investigación se esfuerza por explorar el ámbito de las tecnologías relevantes para entornos de campus inteligentes, con un énfasis específico en el diseño de un marco integral personalizado. El marco previsto pretende actuar como catalizador de la innovación y la excelencia operativa, impulsando así la evolución de los campus inteligentes y contribuyendo a una transformación más amplia de la educación superior. Un elemento central de este esfuerzo es la resolución de desafíos pertinentes, como la escalabilidad, la participación del usuario y la integración perfecta con los marcos existentes. Los objetivos descritos abarcan una revisión exhaustiva de la literatura, el desarrollo de un modelo arquitectónico sólido, la creación de una aplicación web diseñada específicamente para abordar las necesidades de tutoría de los estudiantes, la integración de capacidades sofisticadas de análisis de datos y el diseño de una interfaz de usuario intuitiva guiada por principios de diseño centrados en el usuario. Al perseguir diligentemente estos objetivos, la investigación busca mejorar la eficiencia, la accesibilidad y la satisfacción del usuario dentro del entorno universitario, sentando una base sólida para la realización de un entorno de campus inteligente, sofisticado y adaptable.

Palabras clave: campus inteligente, desarrollo web moderno, framework web

Abstract

In recent years, a notable shift has been observed in the educational landscape, evolving from traditional digital campuses to sophisticated ecosystems of smart campuses. This evolution underscores the critical need to leverage advanced technologies to optimize operational efficiency and service delivery within academic institutions. Despite strides made by Ecuador in fortifying its educational technology infrastructure, the establishment of smart campuses remains in its nascent phase, necessitating concerted efforts to align with global educational trends. This research endeavors to explore the realm of technologies relevant to smart campus environments, with a specific emphasis on devising a comprehensive framework tailored. The envisioned framework aims to act as a catalyst for innovation and operational excellence, thereby driving the evolution of smart campuses and contributing to the broader transformation of higher education. Central to this endeavor is the resolution of pertinent challenges such as scalability, user engagement, and seamless integration with existing frameworks. The outlined objectives encompass an exhaustive literature review, the development of a robust architectural blueprint, the creation of a purpose-built web application addressing student tutoring needs, the integration of sophisticated data analytics capabilities, and the design of an intuitive user interface guided by user-centric design principles. By diligently pursuing these objectives, the research seeks to enhance efficiency, accessibility, and user satisfaction within the university milieu, laying a solid foundation for the realization of a sophisticated and adaptive smart campus environment.

Keywords: smart campus, modern web development, web framework

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

With the rapid advancement of information technology, universities worldwide are continually evolving their management strategies to enhance operational efficiency. One significant transformation observed in recent years is the transition from traditional digital campuses to intelligent campus environments [2]. This shift underscores the importance of leveraging cutting-edge technologies to optimize campus operations and services. While the concept of digital campuses has played a crucial role in enhancing information management within educational institutions, the emergence of smart campuses represents a new frontier in campus development. In Ecuador, the evolution towards smart campuses is still in its nascent stages, reflecting the country's ongoing efforts to bolster its educational technology infrastructure. In tandem with global trends, Ecuador has made significant strides in enhancing its education technology landscape, particularly with the adoption of cloud computing, big data, and artificial intelligence. However, there remains a pressing need to accelerate the development of smart campuses to align with the evolving educational landscape.

This research endeavors to address this gap by presenting an in-depth exploration of the technologies relevant to smart campus environments. By analyzing the specific requirements and challenges faced in Ecuador's educational context, this study aims to design a comprehensive framework for a web application tailored to the needs of smart campuses. Through this framework, the aim is to catalyze the advancement of smart campuses in Ecuador, fostering innovation, efficiency, and effectiveness in educational management and service delivery. By harnessing the potential of emerging technologies, such as web applications, this research seeks to contribute to the ongoing transformation of higher education institutions into dynamic, technology-driven hubs of learning and innovation.

1.2 Problem statement

The development of web applications tailored for smart campuses faces several obstacles that hinder their ability to enhance the educational environment. One significant challenge revolves around scalability [3], as these applications must be capable of accommodating increasing user demands as the campus population grows. Ensuring that the infrastructure can handle surges in usage without compromising performance is essential for delivering a seamless user experience and maintaining operational efficiency. Another critical concern is cybersecurity. With the proliferation of sensitive data stored within smart campus systems, protecting against cyber threats is paramount. Any breach in security could lead to the compromise of personal and confidential information, posing significant risks to students, faculty, and administrative staff [4]. Therefore, robust measures must be implemented to safeguard data integrity and maintain user trust in the system. Capturing the attention and engagement of users is also a key challenge in the development of web applications for smart campuses. User-friendly and accessible designs are crucial for encouraging adoption and ensuring that users can navigate the system effortlessly. Interfaces should be intuitive and responsive, catering to diverse user needs and preferences. Failure to prioritize usability could result in decreased user satisfaction and hinder the overall effectiveness of the application. Additionally, seamless integration with existing campus frameworks presents a formidable obstacle. Smart campuses often rely on a complex network of interconnected systems and infrastructure. Ensuring compatibility and interoperability between new web applications and legacy systems is essential for minimizing disruption and maximizing efficiency. Any discrepancies or inconsistencies in integration could lead to operational inefficiencies and hinder the realization of a truly integrated smart campus environment.

Addressing these challenges is essential for creating a digital environment that meets the evolving needs of students, faculty, and administration. By overcoming obstacles related to scalability, cybersecurity, usability, and integration, developers can lay the foundation for a truly smart campus that enhances learning outcomes, fosters innovation

1.3 Objectives

1.3.1 General Objective

Develop an integrated framework for web application development optimized for smart campuses, emphasizing scalable architecture, user-friendly interfaces, and data analytics capabilities to enhance efficiency, accessibility, and user experience within university environments.

1.3.2 Specific Objectives

1. Conduct a comprehensive literature review on the topic of smart campuses to analyze existing research, identify key trends, and gain insights into the current state of knowledge in this field.

2. Develop an scalable architecture as a framework for web application development that supports the dynamic and diverse needs of smart campuses, emphasizing adaptability and user experience.

3. Develop a web application aimed at facilitating student-to-student tutoring requests, utilizing the Smart Campus model to enhance accessibility, efficiency, and user experience in managing tutoring services within the university environment, serving as a proof of concept for the proposed framework.

4. Design a data integration component for the proposed framework aimed at enhancing data analytical capabilities to enable data-driven decision-making processes.

5. Design a user interface for the web application that is attractive and accessible, leveraging user-centered design principles to foster positive interactions and satisfaction among students, faculty, and administrative staff.

Chapter 2

Theoretical Framework

In recent years, the concept of a smart campus has emerged as transformative in educational environments. As technology permeates our lives, institutions drive digital innovations to create interconnected ecosystems known as smart campuses. This chapter delves into smart campus conceptualization, exploring defining characteristics. It addresses core principles of web applications, including client-server architecture and database management. Additionally, it emphasizes user-centered design's importance in crafting user-friendly interfaces, and agile methodologies' benefits in software development within smart campuses.

2.1 Smart Campus Conceptualization

Since modern technology such as computers and the internet has flooded our society, the digital era has brought us concepts like "digital literacy" as a guide to survive in this technological world [5], along with various studies on the imminent changes at that time and how to adapt to them in various areas such as marketing [6], teaching methods [7], or even the way governments operate [8]. The word "smart" has become a popular term in recent decades, as seen in smartphones where a device has evolved from being simply for calls and messages to having added functionalities. Another example is smart bulbs with more features than just turning on and off. The "smart" concept has also extended to interconnected ecosystems much larger than a single device, such as smart cities and smart houses. In this section, we will direct our focus towards one particular ecosystem: the smart campus.

2.1.1 Defining Smart Campuses

Numerous researchers have contributed to the study of smart campuses, offering various viewpoints and definitions [9, 10, 11, 12, 13, 2, 14]. Generally, the idea of a smart campus is approached from two main angles: a focus on technology and a broader perspective influenced by smart city concepts. The technology-centered approach emphasizes how technologies like the Internet of Things (IoT) and Information and Communication Technologies (ICT) play vital roles in modernizing institutional services [9, 10, 11, 12]. On the other hand, some scholars view smart campuses through the lens of smart city principles, aiming to enhance various aspects of campus life, including economy, governance, and quality of life, through advanced technologies. For example, Paola et al. (2019) in [13] describe a Smart Campus as a digitally enhanced environment where sensors are widely deployed to respond dynamically to users' activities and environmental changes. This vision includes integrating information systems, digital connectivity, and potentially digital interfaces to improve the overall campus experience. It highlights the transformative potential of technology in creating more adaptable and intelligent educational environments. In another study, Chagnon-Lessard et al. (2021) [2] suggest that smart campuses serve as ideal settings for innovation and technological experimentation due to their strategic position at the intersection of research and innovation. Educational institutions like universities and colleges, being hubs of research and technological advancement, play critical roles in developing and implementing intelligent systems. However, they must navigate the unique complexities of campus environments, considering diverse occupants and specialized infrastructure such as laboratories [14]. This requires adopting tailored smart solutions to address the specific needs of advanced educational settings effectively.

2.1.2 Smart Campus Ecosystem

The concept of a smart campus extends beyond mere technological application within an academic setting; it encapsulates a dynamic and interconnected ecosystem that converges various technological and human elements to enhance overall efficiency, sustainability, and user experience [14]. At the core of the smart campus ecosystem lies a robust technological infrastructure. This infrastructure encompasses a network of interconnected sensors, devices, and systems that collect and disseminate data. These data sources range from environmental sensors monitoring air quality and energy consumption to smart devices used by students and faculty. The integration of this diverse technological landscape forms the foundation for creating intelligent applications that respond to the needs of the campus community [15]. The smart campus operates as a networked environment where seamless interconnectivity and data flow are essential. Information generated by various sensors and devices should be efficiently transmitted and processed to derive meaningful insights [3]. This interconnectedness facilitates real-time decision-making, resource optimization, and the creation of personalized experiences [16]. Given these dynamics, exploring the flow of data within the smart campus ecosystem becomes imperative for crafting web applications that leverage this information effectively.

2.1.3 Smart Campus Benefits

The adoption of smart campus environments brings a range of advantages that extend beyond the traditional functions of academic institutions. These benefits can be grouped into various key areas, highlighting the transformative influence of infusing technology into the educational landscape.

1. **Streamlined Operational Efficiency**: Smart campuses harness technology to simplify administrative tasks, optimize resource distribution, and automate routine functions. This leads to improved operational efficiency, enabling educational institutions to allocate resources more efficiently and focus on core educational goals [17].

2. **Sustainability and Environmental Impact**: The inclusion of environmental sensors and sustainable practices in the smart campus ecosystem contributes to a more eco-conscious and sustainable atmosphere. Through real-time monitoring of energy usage, waste management, and resource utilization, smart campuses can minimize their environmental impact and endorse environmentally responsible practices [18].

3. **Enhanced User Experience**: The user experience in a smart campus is significantly enriched through tailored services and seamless interactions. Smart applications designed for students, faculty, and staff enhance communication, accessibility, and overall satisfaction, fostering a positive academic environment [14].

4. **Data-Driven Decision Making**: The wealth of data generated by sensors and devices within the smart campus provides valuable insights for informed decision-making. Institutions can analyze patterns, recognize trends, and make informed choices to enhance the overall effectiveness of educational processes and administrative strategies [16].

5. **Collaborative Learning and Research**: Smart campuses promote collaborative learning environments by providing tools and platforms that facilitate smooth communication and knowledge sharing. This interconnected ecosystem encourages interdisciplinary research and collaborative initiatives, nurturing a culture of innovation within academic communities [19].

6. **Adaptability to Technological Advances**: A smart campus is inherently adaptable to technological advancements, ensuring that the educational environment stays at the forefront of innovation. This adaptability allows institutions to integrate emerging technologies and stay current with the evolving needs of the academic community and industry [20].

The achievement of these benefits depends on the comprehensive and well-coordinated integration of technology, human factors, and data-driven insights within the smart campus ecosystem. As we delve deeper into the theoretical framework, it becomes evident that these advantages provide the basis for developing web applications specifically designed to meet the unique needs of a smart campus environment.

2.2 Core principles of Web Applications

The development of web applications relies on foundational architectural principles and user-centered design approaches to create intuitive and efficient digital experiences. In this section, we explore key concepts related to the client-server architecture, front-end and back-end development, database management, and user-centered design (UCD). Understanding these concepts is essential for designing and implementing effective web applications that meet user needs and enhance overall usability and satisfaction.

2.2.1 Client-Server Architecture

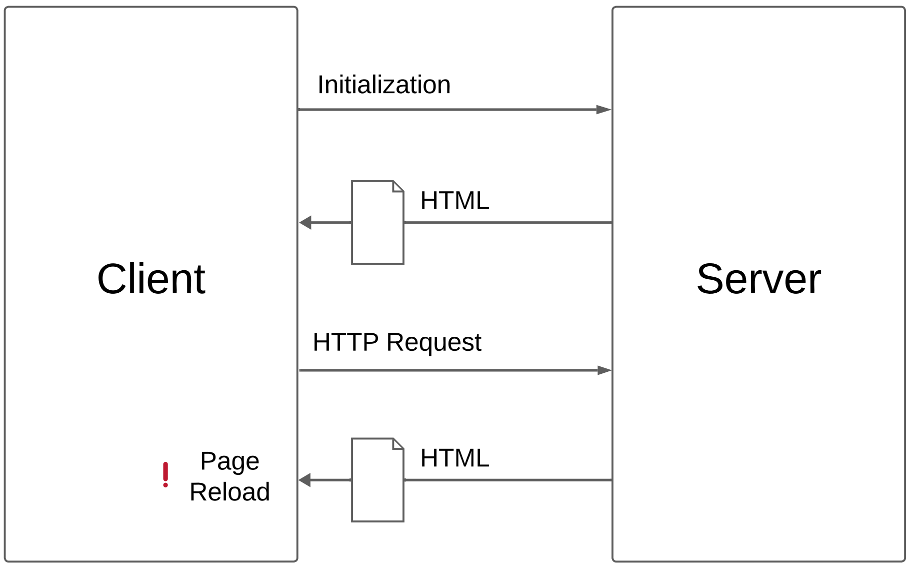


Figure 2.1: Client-Server architecture lifecycle.

The client-server architecture is fundamental to web applications, facilitating communication between multiple clients (typically users' devices with web browsers) and a central server (see Figure 2.1). The server serves as the host, processor, and deliverer of web applications to clients, handling requests, executing back-end logic, and managing data storage and retrieval. In contrast, the client, often a web browser, sends requests to the server, receives responses, and presents them to the user. This communication occurs over HTTP/HTTPS protocols, with clients utilizing HTTP methods (GET, POST, PUT, DELETE) to interact with the server. Notably, each interaction between client and server is stateless, meaning the server does not retain client state between requests. However, technologies such as cookies, session storage, and databases are employed to maintain statefulness as needed. Additionally, scalability is a crucial consideration in web application development. Techniques like load balancing, caching, and distributed database systems are implemented to optimize performance and ensure scalability, accommodating varying levels of user demand effectively.

2.2.2 Front-End concepts

Front-end development centers on creating the user interface and experience of web applications. It encompasses everything users interact with directly in their web browsers. The primary languages used in front-end development are HTML for structure, CSS for styling, and JavaScript for functionality. Frameworks like React, Angular, and Vue.js significantly enhance development efficiency and user experience [21]. Additionally, Responsive Design ensures web applications work seamlessly across various devices and screen sizes, utilizing fluid grids, flexible images, and CSS media queries [22].

2.2.3 Back-End concepts

Back-end development focuses on the server, database, and application logic. It involves using server-side programming languages such as Python, Ruby, Node.js, or Java, and frameworks like Django, Rails, or Express. The back-end is responsible for managing database interactions, developing APIs, user authentication, and executing server-side logic. Modern approaches like microservices architecture and RESTful APIs are widely adopted for building scalable and maintainable back-end systems. Furthermore, back-end development must prioritize security measures, including data protection, secure API endpoints, and mitigating common threats like SQL injection and cross-site scripting [23].

2.2.4 Database Management

Database management encompasses the processes of storing, retrieving, updating, and managing data. Commonly used databases include relational databases such as MySQL and PostgreSQL, and non-relational databases like MongoDB. Key concepts in this field include database design, normalization, and SQL for relational databases. NoSQL databases provide flexibility for specific types of data and use cases. Ensuring data integrity, implementing robust security measures, maintaining regular backups, and optimizing query performance are essential components of effective database management [24].

2.3 User-Centered Design

User-Centered Design (UCD) is a set of principles that place users at the center of the design process. These principles were first established through research on Human-Computer Interaction in 1987, emphasizing the importance of considering the needs, preferences, and limitations of end-users when creating products that are usable and accessible to as many people as possible [25].

2.3.1 User-Centered Design Concepts

UCD interaction can take many forms; some approaches involve regular user consultations to understand their needs and apply their suggestions during crucial phases such as requirements collection and usability testing [26]. Norman later explained that the utility and comprehensibility of products are achieved when users understand what actions to take and can comprehend what is happening [27]. Essentially, the core of user-centered design is tailoring products to meet user expectations and needs, ensuring an intuitive experience where users can comprehend and perform necessary actions effectively. According to Norman, understanding and usability are key to the success of user-centered design, ensuring the product not only fulfills required functions but also facilitates interaction without causing confusion.

Two key practices associated with UCD are:

- **Usability Testing:** In line with the UCD model, usability testing involves real users actively participating in evaluating a developing system or product to ensure it meets defined usability criteria. According to Dumas and Redish (1993), usability testing is a systematic method for observing users as they interact with a product [28]. This process collects detailed information on how users perceive the ease or difficulty of the product, providing valuable feedback on the user experience. This feedback allows for iterative adjustments and continuous improvements in design to optimize usability and ensure effective and satisfactory interaction with the system.

- **Participatory Design:** Unlike usability testing, participatory design involves user involvement from the initial design stages to final implementation. It is crucial to integrate employees' skills and experiences into the organizational development and execution of computer systems and related tasks [29]. Participatory Design (PD) seeks to continuously incorporate the user's perspective, fostering co-creation and shared decision-making. PD researchers explore the conditions that promote active user participation in the design and implementation of computer systems in the workplace [30].

2.3.2 UCD Process

According to Keinonen (2008), UCD processes focus on users throughout the planning, design, and development of a product [31]. UCD can be divided into several stages to meet the needs and constraints of end-users. According to Preece et al. (2002) as cited in [26], there are seven phases that encompass the design and development cycle of a product:

- **Background Interviews and Questionnaires:** Conducted in the early stages of the design project to gather data on user needs and expectations. This involves evaluating design alternatives, prototypes, and the final artifact before its development.
- **Sequence of Work Interviews and Questionnaires:** Applied in the initial phase of the design cycle to gather fundamental data on the sequence of tasks that users will perform.
- **Focus Groups:** Convene a diverse range of stakeholders to discuss issues and requirements associated with the design. This methodology is implemented early in the design cycle to capture a variety of perspectives and opinions, enriching the creative process.
- **On-site Observation:** Involves observing the use of the product or system in its actual environment to gain a deep understanding of the context in which users will interact with the designed artifact. Detailed data on environmental factors, physical conditions, and relevant elements influencing usability and user experience are collected during this stage.
- **Role Playing, Walkthroughs, and Simulations:** Conducted both at the beginning and midpoint of the design cycle, this involves designers and users participating in simulated usage scenarios, interacting with prototypes or specifically designed scenarios.
- **Usability Testing:** A crucial phase in the design cycle, located in the final stage, where quantitative data is collected. This process focuses on evaluating measurable usability criteria, providing objective information on the performance and effectiveness of the designed artifact.
- **Interviews and Questionnaires:** Takes place in the final stage, where qualitative data is collected. This process focuses on obtaining detailed information about

user satisfaction with the designed artifact, seeking a deep understanding of their perceptions, opinions, and experiences.

Each of these stages is essential for creating a product that truly meets user needs and expectations. By systematically involving users throughout the design and development process, UCD ensures that the final product is both functional and user-friendly. This iterative approach allows designers to identify and address potential issues early, making necessary adjustments based on user feedback. As a result, the end product not only achieves its intended purpose but also provides a positive and intuitive user experience, ultimately leading to higher user satisfaction and adoption rates.

2.3.3 UCD in the Educational Context

The implementation of emerging technologies in education was a prominent topic during the 1990s. However, the implications of UCD in educational contexts were explored as early as 1996, emphasizing the importance of UCD in hypertext and hypermedia for learning [32]. The application of instructional design strategies for students is analogous to user-centered approaches aimed at creating interfaces. Therefore, applying UCD in education recognizes the importance of considering the student as the primary user, allowing for continuous improvement in the quality of instruction through adaptation and attention to the individual needs of students [33]. In the educational context, UCD involves carefully considering the individual characteristics of students, such as their level of cognitive development, maturity, age, skills, limitations, interests, opinions, and knowledge.

2.4 Data Integration and Analytics

2.4.1 Data Integration Strategies

Data integration aims to combine data from diverse sources to create a unified, comprehensive view. This process often involves the use of Extract, Transform, Load (ETL) methodologies [34], where data is extracted from various sources, transformed for consistency, and then loaded into a central repository like a data warehouse [35]. Middleware solutions such as Enterprise Service Buses (ESB) [36] facilitate seamless communication and data exchange between different systems. In the era of service-oriented architectures, API-based integration has become particularly useful, offering a flexible and scalable way to connect disparate software components [37]. Additionally, data virtualization techniques [38] provide real-time or near-real-time views of data from multiple sources without requiring physical integration. A variant of this, data federation [39], creates a virtual database for integrated access and manipulation of data, enabling a consolidated view across an organization.

2.4.2 Data-Driven Decision-Making

Data-driven decision-making focuses on basing decisions on data analysis rather than solely on intuition. This process begins with the collection and management of relevant data from various sources [40]. The core of this approach is the rigorous analysis and interpretation of this data, utilizing statistical tools to derive actionable insights. These insights are then communicated through intuitive reporting and visualization tools like dashboards, which guide decision-making processes [41]. Predictive analytics [42], using advanced algorithms and machine learning, is increasingly employed to anticipate future trends and behaviors. This approach is dynamic, with a continuous feedback loop that updates and refines strategies based on new data and insights, thereby enhancing the overall decision-making process.

2.5 Agile Methodologies

Agile methodologies are a set of principles and practices for software development that prioritize flexibility, collaboration, customer feedback, and rapid iteration. Originating from the Agile Manifesto in 2001 [43], these methodologies prioritize direct communication over documentation, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. Empirical studies [44, 45] support the effectiveness of Agile methodologies in improving project success rates, team productivity, and customer satisfaction. Key principles of Agile include customer satisfaction through early and continuous delivery, welcoming changing requirements, frequent delivery of working software, close daily cooperation between business people and developers, building projects around motivated individuals, face-to-face conversation as the best form of communication, working software as the primary measure of progress, sustainable development pace, continuous attention to technical excellence, simplicity, and self-organizing teams [46]. Two popular Agile methodologies are:

- **Scrum:** A prominent Agile framework, Scrum is structured around fixed-length iterations called Sprints, typically lasting two to four weeks. It emphasizes teamwork, accountability, and iterative progress towards a well-defined goal. Scrum roles include the Product Owner, Scrum Master, and Development Team, each with specific responsibilities to ensure efficient project progression [47].

- **Kanban:** Another Agile approach, Kanban, focuses on visualizing the workflow and limiting work-in-progress to enhance efficiency. It involves using a Kanban board, where tasks are moved from “To Do” to “Done”, providing a transparent overview of the project’s current state. This method is particularly useful for managing ongoing projects with continuously changing or emerging requirements [48].

While both Scrum and Kanban are Agile methodologies and share common principles like flexibility, adaptability, and continuous improvement, they differ in structure and focus. Scrum is more structured, with defined roles and time-boxed sprints, making it ideal for projects requiring rigorous management and where changes are less frequent during each sprint. Kanban, on the other hand, offers more flexibility in terms of workflow management. It is suitable for projects with continuous changes or maintenance tasks, as it allows for a more fluid task management without the confines of fixed sprints. This flexibility makes Kanban a better choice for projects with evolving or unpredictable requirements, while Scrum is more suited for projects where goals and deliverables can be clearly defined in stages. Agile methodologies are particularly relevant in dynamic project environments like those encountered in smart campus development, where requirements can change rapidly and involvement of various stakeholders is crucial for the project's success.

Overall, this theoretical framework provides a comprehensive understanding of smart campuses, web application development principles, and agile methodologies, laying the groundwork for the subsequent chapters' practical implementations and case studies.

Chapter 3

State of the Art

This chapter offers a comprehensive overview of the current landscape in the field of smart campuses. It begins with an exploration of various initiatives, including Smart Campus Technologies & Services, detailing their approaches and applications. It also presents a detailed examination of a Smart Campus Framework based on the findings of literature, highlighting the main enablers and technological frameworks necessary for this transition. The insights provided in this chapter set the stage for the development of a comprehensive web development framework tailored for smart campus environments.

3.1 Smart Campus Initiatives

In exploring the potential of IoT for creating intelligent environments in educational settings, the authors in [49] presents his Smart Campus, a case study in the University of Bologna (Italy) demonstrating the integration of sensor networks and data visualization in a university campus. This system comprises a sensor infrastructure for real-time data collection and a web-based application for interactive data engagement. The involvement of 135 students in its evaluation revealed a significant interest in contributing to the system's development, highlighting the feasibility and appeal of IoT in enhancing interactive and participatory educational environments. This study underlines the transformative impact of IoT in making educational spaces more responsive and student-centered, pointing towards a future of increased student involvement in shaping their learning environments.

In [50], Verstaevel et. al, (2017) describe the implementation of the Smart Campus technique at the University of Toulouse III Paul Sabatier, turning it into an "in vivo" incubator for Smart and Sustainable Cities. The neOCampus operation, initiated in 2013, aims to transform the university into an experimental environment for improvements in quality of life, reduction of ecological footprint, and efficiency in operating costs. The application of interdisciplinary initiatives is highlighted, such as an open data platform, ecocitizenship projects to raise awareness about the ecological footprint, and the optimization of energy consumption in classrooms through machine learning systems. Additionally, biodiversity is addressed through a participatory application that identifies and locates fauna and flora on campus, facilitating data collection to study the impact of human activities during the urbanization process. These projects demonstrate the need for interdisciplinary collaboration to address the challenges and opportunities associated with Smart Cities.

The UMS HopIn! application [51] represents an initiative towards the implementation of a smart campus at the University of Malaysia Sabah (UMS), specifically in Borneo. Designed as a real-time bus tracking system, it consists of two mobile applications, UMS HopIn! and UMS HopIn! Driver, along with the web application UMS HopIn! Admin. This application aims to address the issue of bus schedule reliability through effective tracking, allowing students to be aware of real-time bus locations and estimated arrival times. User experience evaluation using the MeCUE technique reveals that the application meets expectations and receives a good UX rating, supporting the development of future smart campus initiatives. Feedback received reinforces the utility and usability of the application, positioning it as an effective tool to enhance campus life and support upcoming smart campus initiatives.

3.2 Smart Campus - Technologies & Services

According to Yang et al. (2018) [52], with the rise and rapid development of cloud computing, big data, and IoT, advanced ICT has gradually been integrated into the education sector, leading to constant improvement in university informatization levels. In this scenario, literature indicates that universities worldwide are implementing smart campus concepts into their academic environments, promoting direct changes in the quality of life of students, researchers, staff, visitors, passersby, and other professionals involved in university management.

In their systematic review [53], Zhang et al. analyze the evolving landscape of smart campus technologies and applications, emphasizing the need for an updated examination in light of recent advancements and the impacts of the COVID-19 pandemic. This article stands out for its comprehensive analysis of current smart campus initiatives, categorizing them across various domains to identify prevailing research trends. Significantly, the study adopts a human-centered approach, aligning with the core principle of smart campus development, which prioritizes the needs and interests of stakeholders. The authors present a detailed case study to assess how well current research aligns with these human-centered objectives. This review is crucial in understanding the heightened importance of smart campuses, especially their role in offering remote, personalized, and ubiquitous services during challenging times. It provides a contemporary snapshot of the field, highlighting the integration of cutting-edge information and communication technologies to augment the efficiency and effectiveness of educational services. Some examples of useful web systems on Smart Campus context are:

- **Learning Management System (LMS):** Web services can be integrated for managing course content, facilitating online learning, tracking student progress, and enhancing teacher-student interaction [54, 55, 56].
- **Campus Navigation and Maps:** Implementing services like Google Maps API for campus navigation helps students and visitors find their way around the campus, including buildings, departments, and amenities [57, 58].
- **Library Management System:** Integrating a web-based library management system enables efficient tracking of books, digital resources, and other materials, facilitating online reservations, renewals, and automated notifications [59, 60, 61, 62, 63].
- **Student Information System (SIS):** This service manages student data, including admissions, enrollment, grades, and schedules. Integrating an SIS helps streamline administrative tasks [64, 65].
- **Campus Safety and Security Services:** Implementing web services for campus safety, like emergency notification systems [66] and mobile safety apps [67], can enhance security and provide quick access to emergency assistance.
- **Transportation Services:** Integrating real-time information about campus shuttles [68], public transit [69], and parking availability [57].
- **Sustainability and Energy Management:** Web services for monitoring and managing energy usage and sustainability initiatives on campus [70, 71].
- **Career Services and Internship Portals:** Platforms connecting students with job opportunities, internships, and career advice [72].
- **IoT Integration for Smart Facilities:** Using web services to integrate IoT devices for smart lighting [73], body temperature control [74], or any environmental monitoring [75] in campus buildings.

Finally, Msitah Mausa et al. [76] state that smart campuses consist of environments capable of providing efficient technologies and infrastructures in service delivery to enhance and support the teaching process, research, and student experience. For Lam-for [77], it is a new paradigm of thought within a holistic environment, encompassing at least, but not limited to various aspects of artificial intelligence [78], such as e-learning, social networks, and communications for collaboration in both academic and administrative work, sustainability, and ICT, with intelligent sensor management systems, medical care, smart building management, with automated security control and transparent governance of campuses. The conceptual approach of smart campuses considers that the spaces or territories on which university campuses are located are managed as cities, requiring effective management models, from secure database sources that facilitate decision-making by managers, and especially, in the case of public universities, transparency becomes one of the elements that configure public governance [79, 80] or university governance [81, 82].

3.3 Smart Campus Enablers

Based on the findings of Vian Ahmed, Karam Abu Alnaaj, and Sara Saboor presented in their comprehensive literature review on Smart Campus criteria in [1], their research, which focuses on the transformation of traditional campuses into Smart Campuses, outlines the main enablers and technological frameworks necessary for this transition.

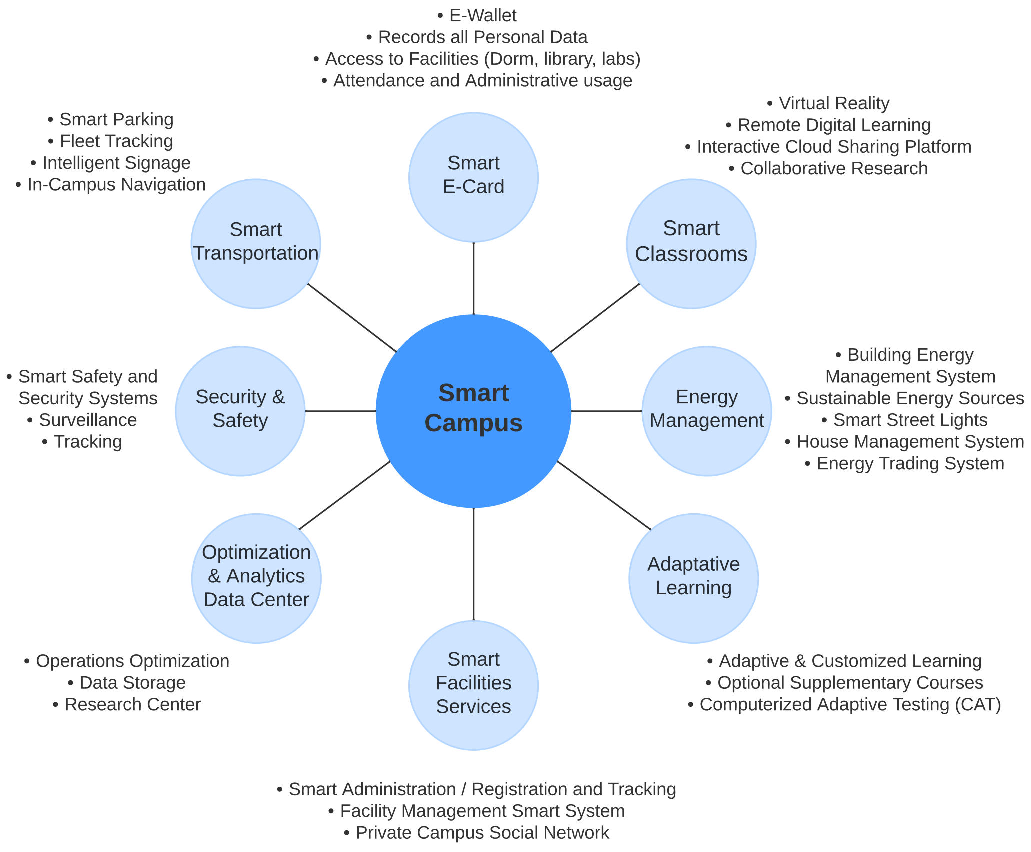

• E-Wallet
• Records all Personal Data
• Access to Facilities (Dorm, library, labs)
• Attendance and Administrative usage
Smart E-Card
• Smart Parking
• Fleet Tracking
• Intelligent Signage
• In-Campus Navigation
Smart Transportation
• Virtual Reality
• Remote Digital Learning
• Interactive Cloud Sharing Platform
• Collaborative Research
Smart Classrooms
Smart Campus
• Smart Safety and Security Systems
• Surveillance
• Tracking
Security & Safety
Energy Management
• Building Energy Management System
• Sustainable Energy Sources
• Smart Street Lights
• House Management System
• Energy Trading System
Optimization & Analytics Data Center
• Operations Optimization
• Data Storage
• Research Center
Adaptative Learning
• Adaptive & Customized Learning
• Optional Supplementary Courses
• Computerized Adaptive Testing (CAT)
Smart Facilities Services
• Smart Administration / Registration and Tracking
• Facility Management Smart System
• Private Campus Social Network

Figure 3.1: Main enablers of a Smart Campus proposed by [1]

Figure 3.1 illustrates the conceptual model of the main enablers for a Smart Campus, designed to integrate advanced technologies and data-driven strategies to enhance the educational environment. This model emphasizes the interaction between various smart components, creating an efficient, sustainable, and user-friendly campus experience.

3.4 Smart Campus Framework

From literature review, a Smart Campus Framework can be defined and its illustrated in Figure 3.2. This framework consists in various layers that integrate technology, data, applications, and policies to create an intelligent and efficient campus environment. This section introduces the general architecture of the Smart Campus Framework, detailing its various layers and their respective components. Each layer plays an important role in transforming a traditional campus into a smart one.

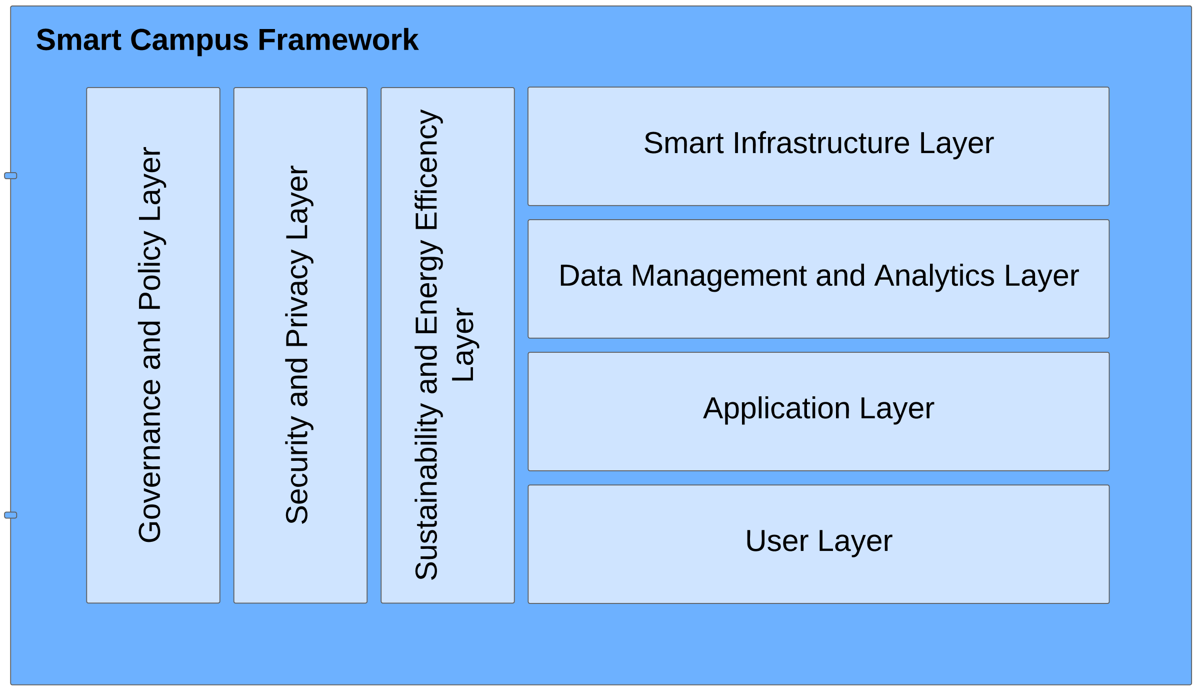

Smart Campus Framework
Governance and Policy Layer
Security and Privacy Layer
Sustainability and Energy Efficency Layer
Smart Infrastructure Layer
Data Management and Analytics Layer
Application Layer
User Layer

Figure 3.2: Smart Campus Framework

3.4.1 Smart Infrastructure Layer

The Smart Infrastructure Layer forms the basis of the Smart Campus framework, ensuring robust connectivity, efficient data management, and the integration of IoT devices.

- **Network Infrastructure:** A reliable and high-speed network infrastructure is essential for supporting the vast array of connected devices and systems within a smart campus [83]. This includes wired and wireless networks, ensuring seamless connectivity and data transfer across the campus.
- **IoT Devices:** Deployment of IoT devices across the campus to monitor and manage environmental conditions, energy consumption, and security. IoT significantly improves the efficiency of campus operations [16] by providing granular data and enabling automated responses to changing conditions.
- **Data Centers and Cloud Services:** These provide the necessary computational power and storage to handle the vast amounts of data generated by IoT devices and smart campus applications. Priorizing cloud services also enhance scalability and flexibility [84].

3.4.2 Data Management and Analytics Layer

This layer focuses on the processes involved in handling and finding insights from the data collected by IoT devices and other sources.

- **Data Collection:** Collecting data from various sources, including IoT devices, user interactions, and administrative systems, provides a view of campus operations and user behavior.
- **Data Processing:** Once collected, data must be processed to remove redundancies, correct errors, and structure it for analysis. Advanced data processing techniques ensure that data is clean, accurate, and ready for analysis.
- **Analytics and Insights:** Using big data analytics [85] and machine learning [86, 87] to retrieve actionable insights from collected data. These insights can helps in data-driven decision-making, optimize operations, and enhance the user experience.

3.4.3 Application Layer

The Application Layer focuses on the various applications and services deployed within a smart campus to enhance academic, administrative, and campus life experiences. Applications and services can be defined in three categories:

- **Academic and Administrative Applications:** These include Learning Management Systems (LMS), Student Information Systems (SIS), and other tools that support educational and administrative processes are fundamental to the smart campus ecosystem .
- **Campus Services:** Applications that manage campus services such as transportation, safety, and facility management fall under this category. These services enhance the efficiency and effectiveness of campus operations.
- **Campus Life Applications:** Applications focused on improving the overall campus experience for students, faculty, and staff. This includes services for campus navigation, event management, and community building.

A more detailed review of this services was developed in section 3.2.

3.4.4 User Layer

The User Layer focuses on the different stakeholders within a smart campus and creating user-centered services and interfaces that enhance their overall campus experience

- **Students:** Smart campus systems provide students with tools and services to enhance their learning experiences and campus life, including access to educational resources, campus services, and community engagement platforms.
- **Faculty and Staff:** Faculty and staff benefit from smart campus systems through the integration of administrative processes, access to teaching resources, and tools for collaboration and communication.
- **Visitors and Guests:** Smart campus systems also fulfils the needs of visitors and guests, providing them with information, navigation assistance, and access to public campus services.
- **Stakeholders Engagement:** Effective engagement strategies such as continuous feedback mechanisms [88] and community awareness [49, 52] ensure that all stakeholders are involved in the development and implementation of smart campus initiatives, promoting a collaborative and inclusive environment.

3.4.5 Security and Privacy Layer

This layer ensures the protection of data, systems and physical assets on campus.

- **Cybersecurity Measures:** Robust cybersecurity measures protect the campus network and data from unauthorized access, cyberattacks, and breaches. This includes firewalls [4], intrusion detection systems [89], and regular security audits.

- **Data Privacy:** Ensuring the privacy of personal and sensitive data is fundamental. Accordance with data protection regulations and implementing privacy policies are essential for maintaining trust and legal compliance [90].

- **Physical Security:** Physical security measures such as surveillance cameras system [91], access control systems [92], and emergency response systems [93, 94] ensure the safety and security of the campus community.

3.4.6 Sustainability and Energy Efficiency Layer

This layer focuses on integrating sustainable practices and energy-efficient technologies on the campus.

- **Renewable Energy Sources:** Incorporating renewable energy sources such as solar panels and wind turbines [95] reduces the campus's carbon footprint and promotes sustainability.

- **Energy Management Systems:** Advanced energy management systems [94, 96] monitor and control energy usage across the campus, ensuring efficient and sustainable operations.

- **Sustainable Practices:** Encouraging sustainable practices across campus operations, including waste reduction, recycling, and sustainable transportation options.

3.4.7 Governance and Policy Layer

The Governance and Policy Layer provides the strategic direction and management required to implement and sustain the smart campus initiatives.

- **Strategic Planning:** Developing a strategic plan that outlines the vision, goals, and objectives for the smart campus initiatives. This includes setting clear objectives, timelines, and performance metrics [20].

- **Policy Development:** Developing policies that support the adoption and use of smart technologies inside the campus. This includes policies on data management, privacy, security, and sustainability.

- **Management and Coordination:** Effective management and coordination of smart campus projects require collaboration among various stakeholders, clear communication channels, and efficient project management practices.

The Smart Campus Framework layers are designed to work together in a cohesive and interconnected manner. The vertical layers (Governance and Policy, Security and Privacy, and Sustainability and Energy Efficiency) provide essential oversight and support that ensure the robust functioning of the horizontal layers (Smart Infrastructure, Data Management and Analytics, Application, and User). After and exhaustive review on smart campus state of the art, the central problem of this research is posed in the following form: How can a framework for web application development be designed and implemented to optimize functionality and user experience in smart campus environments? In this sense, this research aims to conduct a comprehensive analysis of existing web application development methodologies and technologies, considering the specific needs and challenges of smart campus environments. By identifying best practices and innovative approaches, this research seeks to pave the way for the effective implementation of intelligent tools in the daily lives of universities, ultimately enhancing the quality and functionality of education in these settings. Additionally the proposed framework aims to solve the gap on defining a clear, modern and generalized framework for delivering web applications for smart campus environments.

Chapter 4

Methodology

This chapter introduces the Framework Development section, which provides a comprehensive guide to designing and implementing a web application development framework for smart campuses. The methodology is structured into several key components that collectively ensure a comprehensive and effective approach to web application development. The chapter also discusses front-end, back-end technologies and additional development tools, providing an in-depth look at the technologies and their roles in building a scalable and efficient web application. This comprehensive approach ensures that the application aligns with the proposed objectives of creating intelligent and responsive campus ecosystems.

4.1 Designing the Framework for Web Application Development on Smart Campus

The development of a web application focused on a Smart Campus environment requires a comprehensive and well-structured framework. This section outlines the design of such a framework, incorporating principles and components necessary to support the dynamic and multifaceted needs of a smart campus. The architecture for this framework is illustrated in Fig. 4.1. This framework incorporates various technological and methodological components that are essential for creating a robust, scalable, and user-friendly application and is structured into a tri-component architecture: application, data and user-centered design.

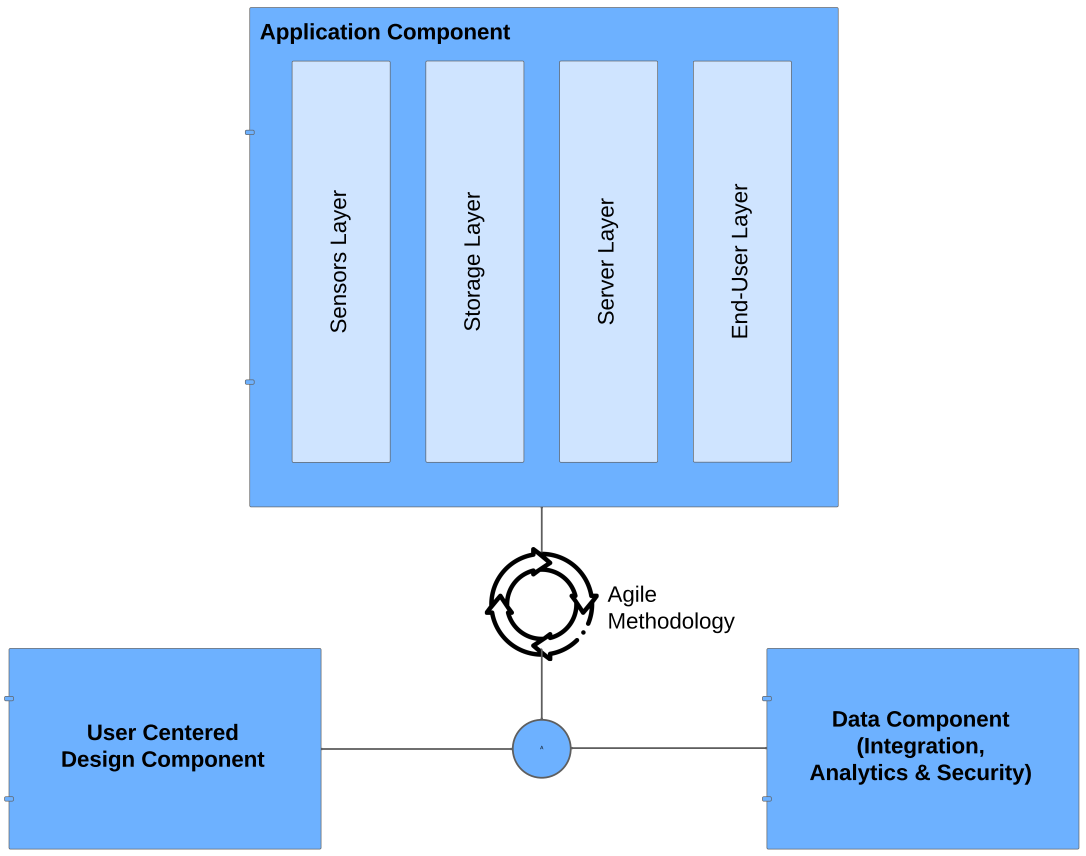


Figure 4.1: Web Application Development Framework for Smart Campus Environments

The framework development process began with a thorough review of the Smart Campus Framework, which consists of multiple layers, including Smart Infrastructure, Data Management and Analytics, Application, User, Security and Privacy, Sustainability and Energy Efficiency, and Governance and Policy. By understanding the interrelations among these layers, the web application framework was designed to seamlessly integrate with the overarching smart campus infrastructure.

4.1.1 Key Elements of the Framework

The essential components that form part of the proposed web application development framework for smart campus environments are:

1. **Application Component:** This component includes the core functionalities of the web application, structured into four layers:

    - **Sensors Layer:** Integrates with IoT devices to gather real-time data from the campus environment.
    - **Storage Layer:** Handles the storage of data collected from various sources, ensuring efficient data management.
    - **Server Layer:** Manages the server-side logic and processes, facilitating smooth operations and data flow.
    - **End-User Layer:** Focuses on delivering an intuitive and user-friendly interface for the end-users, including students, faculty, and staff.

2. **Agile Methodology:** Central to the framework, agile methodologies are employed to facilitate iterative development, continuous feedback, and rapid adaptation to changing requirements.

3. **User-Centered Design Component:** Emphasizes the importance of designing the application with the user in mind, ensuring that the interface is accessible, intuitive, and meets the diverse needs of all campus stakeholders.

4. **Data Component (Integration, Analytics & Security):** This component is responsible for integrating various data sources, performing analytics to derive actionable insights, and ensuring the security of data throughout its lifecycle.

4.1.2 Alignment with the Smart Campus Framework

How the proposed web application development framework aligns with the existing Smart Campus Framework:

- **Smart Infrastructure Layer:** The Sensors Layer and Storage Layer of the application component directly align with the Smart Infrastructure Layer by ensuring robust connectivity, efficient data management, and the integration of IoT devices.

- **Data Management and Analytics Layer:** The Data Component of the framework focuses on data integration, analytics, and security, mirroring the functions of the Data Management and Analytics Layer in the Smart Campus Framework.

- **Application Layer:** The server and end-user layers within the Application Component correspond to the Application Layer of the Smart Campus Framework, enhancing academic, administrative, and campus life experiences through various applications and services.

- **User Layer:** The User-Centered Design Component aligns with the User Layer, focusing on creating user-centered services and interfaces that enhance the overall campus experience.

- **Security and Privacy Layer:** The emphasis on security within the Data Component ensures alignment with the Security and Privacy Layer, protecting data and systems on campus.
- **Sustainability and Energy Efficiency Layer:** By promoting efficient data management and operations, the framework indirectly supports the Sustainability and Energy Efficiency Layer.

By integrating these considerations, the Web Application Development Framework provides a comprehensive blueprint for developing web applications that not only enhance operational efficiency and service delivery within a smart campus but also foster a secure, sustainable, and user-centric environment.

This structured approach ensures that the developed web application is well-equipped to handle the complexities of smart campus ecosystems, driving innovation and excellence in educational management and student engagement. Each of the three main components: Application Component, User-Centered Design Component, and Data Component (Integration, Analytics & Security) will be detailed in the following sections.

4.2 Application Component

The Application Component is a central element of the Web Application Development Framework for Smart Campus Environments. It is designed to encompass the core functionalities and services of the web application, ensuring a seamless and efficient user experience. This component is structured into four interconnected layers: Sensors Layer, Storage Layer, Server Layer, and End-User Layer. Each layer plays a critical role in the overall functionality and performance of the web application.

4.2.1 Sensors Layer

The Sensors Layer acts as the foundation of the framework, capturing real-time data from various IoT devices spread across the campus. These devices include environmental sensors, motion detectors, and smart meters, which provide continuous streams of data, enabling dynamic interactions and responses within the smart campus.

- **Integration with IoT Devices:** The framework supports the integration of a wide range of IoT devices, ensuring comprehensive data collection from different sources.
- **Data Collection and Transmission:** Sensors continuously collect data and transmit it to the central storage system for processing and analysis.
- **Real-Time Monitoring:** Enables real-time monitoring of campus conditions, enhancing operational efficiency and responsiveness.

Key technologies and protocols commonly used in this layer include:

- **IoT Devices:** Sensors for environmental monitoring, energy usage, or motion detection.
- **Data Acquisition:** Protocols like MQTT (Message Queuing Telemetry Transport), a lightweight messaging protocol for small sensors and mobile devices, optimized for high-latency or unreliable networks.

4.2.2 Storage Layer

The Storage Layer is responsible for efficiently managing the data collected from the sensors or user interactions. This layer utilizes advanced storage solutions to ensure data integrity, security, and accessibility, facilitating seamless data retrieval and manipulation.

- **Data Management:** Implements robust data management practices to handle large volumes of data from various sources.
- **Data Security:** Ensures the security and integrity of stored data through encryption, access controls, and regular backups.

Technologies and methodologies commonly used include:

- **Cloud Storage Solutions:** Services like AWS S3, Google Cloud Storage, and Azure Blob Storage provide scalable storage options.
- **Database Systems:** Utilizes advanced database systems, such as SQL (e.g., MySQL, PostgreSQL) and NoSQL (e.g., MongoDB, Cassandra) databases, to store and manage structured and unstructured data.
- **Data Warehousing Solutions:** Amazon Redshift or Google BigQuery for aggregating and analyzing large datasets.

4.2.3 Server Layer

The Server Layer serves as the processing core of the framework, where data from the storage layer is analyzed, processed, and prepared for presentation. This layer employs robust server-side technologies to handle complex computations, data integration, and logic processing.

- **Server-Side Logic:** Handles the core application logic, processing user requests, executing business rules, and managing data transactions.

- **APIs and Middleware:** Utilizes APIs and middleware to enable seamless communication between different components of the application.
- **Scalability and Performance:** Ensures that the server infrastructure can scale to handle increasing user demands without compromising performance.

Key technologies and practices include:

- **Server-Side Frameworks:** Utilizing frameworks such as Node.js, Express, and Next.js to build scalable and efficient server-side applications.
- **Microservices Architecture:** Implementing microservices to break down the application into smaller, manageable services, each responsible for specific functionality.
- **Serverless Computing:** Implementing serverless platforms like AWS Lambda, Azure Functions, and Google Cloud Functions to handle backend logic without managing server infrastructure.
- **Security:** Utilizing authentication and authorization methods based on JWT or OAuth to ensure secure interactions for users.
- **Caching Mechanisms:** Tools like Redis or Memcached are used to reduce load times and server response delays.

4.2.4 End-User Layer

The End-User Layer focuses on delivering an intuitive and user-friendly interface for the various stakeholders of the smart campus, including students, faculty, staff, and visitors. This layer ensures that users can easily interact with the application, access services, and perform their tasks efficiently.

- **User Interface Design:** Employs user-centered design principles to create an interface that is visually appealing, easy to navigate, and responsive.
- **Accessibility:** Ensures that the application is accessible to users with diverse needs, following best practices for accessibility and inclusivity.
- **User Engagement:** Provides features and functionalities that enhance user engagement, such as personalized dashboards, notifications, and feedback mechanisms.

Technologies and methodologies include:

- **User Interface Prototyping Tools:** Figma or Sketch for designing and iterating on UI components.

- **Front-End Frameworks:** Utilizing React.js for building dynamic user interfaces and Next.js for server-side rendering and static site generation.
- **Responsive Design:** Ensuring the application is accessible on various devices and screen sizes using CSS frameworks like Tailwind CSS and Bootstrap.
- **Accessibility Standards:** Adhering to WCAG (Web Content Accessibility Guidelines) to make the application accessible to users with disabilities.

The Application Component integrates various layers to provide a cohesive and efficient application. By effectively managing data collection, storage, processing, and user interaction, this component ensures that the web application meets the needs of a smart campus environment, driving operational efficiency and enhancing the user experience. The subsequent sections will detail the User-Centered Design Component and the Data Component (Integration, Analytics & Security), further elaborating on the comprehensive framework for developing web applications in smart campus environments.

4.3 User-Centered Design Component

The User-Centered Design (UCD) Component is a fundamental aspect of the Web Application Development Framework for Smart Campus Environments. This component ensures that the development of applications is intrinsically aligned with the needs, preferences, and behaviors of its users. The integration of UCD principles throughout the development lifecycle aims to produce applications that are not only functional and efficient but also intuitive and universally accessible.

4.3.1 Engaging with Stakeholders

A pivotal aspect of integrating UCD principles is the active engagement with stakeholders, encompassing not just the end-users but also anyone affected by the application, including administrators, support staff, and external partners. This engagement begins with comprehensive user research to understand the diverse needs and challenges faced by all stakeholders. Techniques such as interviews, surveys, focus groups, and ethnographic studies are employed to gather rich insights into the user's world.

4.3.2 Usability Testing

Usability testing plays a critical role in the UCD process, providing empirical evidence on how real users interact with the application. This involves observing users as they complete predefined tasks within the application, identifying any usability issues that arise. The insights gained from usability testing are invaluable for refining the user interface (UI) and user experience (UX) design, ensuring the application is both intuitive and effective in meeting user needs.

4.3.3 Iterative Design

The framework adopts an iterative design approach, which is fundamental to UCD. This process involves the creation of prototypes, ranging from low-fidelity sketches to high-fidelity interactive models, that are continually tested and refined based on user feedback. Each iteration seeks to improve the UI/UX based on specific insights gained from user interactions, making the application more user-friendly and accessible with each cycle.

4.3.4 Empathy in Design

A core tenet of UCD is empathy, the ability to understand and share the feelings of the user. Design decisions are made with a deep understanding of the users' emotions, contexts, and challenges, ensuring that the application is not only easy to use but also resonates with users on a personal level. Empathy maps and persona creation are tools commonly used to foster this understanding.

4.3.5 Accessibility Standards

The framework places a strong emphasis on adhering to accessibility standards, ensuring that applications are usable by people with a wide range of abilities and disabilities. This commitment to accessibility involves following established guidelines, such as the Web Content Accessibility Guidelines (WCAG), and conducting accessibility audits to identify and rectify barriers that could prevent users from fully engaging with the application.

4.3.6 Feedback Loops

Finally, the incorporation of feedback loops is crucial for the continuous improvement of the application. Feedback from users is actively sought and meticulously analyzed throughout the development process and beyond. This ongoing dialogue with users ensures that the application evolves in response to changing user needs and preferences, thereby maintaining high levels of user satisfaction and engagement.

By embedding UCD principles into its foundation, the smart campus web application development framework aspires to create applications that are not only technically robust but also deeply connected to the human experience. This approach ensures that the developed applications are well-received, widely adopted, and genuinely enhance the campus experience for all users. The subsequent sections will detail the Data Component (Integration, Analytics & Security), further elaborating on the comprehensive framework for developing web applications in smart campus environments.

4.4 Data Component

The Data Component includes data integration, analytics, and security, ensuring that the vast amounts of data generated across the campus ecosystem are seamlessly integrated, processed, analyzed, and securely managed. This component is divided into three subsections: Data Integration, Data Analytics, and Data Security.

4.4.1 Data Integration

Data integration aims to combine data from diverse sources to create a unified, comprehensive view. This process involves the use of various methodologies and technologies to ensure that data is consistently and efficiently integrated.

- **Seamless Data Integration:** The framework ensures that data from various sources, including academic records, campus facilities, IoT devices, and user interactions, are integrated in a seamless manner. This involves the establishment of a robust technical infrastructure that supports the aggregation of real-time and historical data, ensuring that it can be accessed and processed efficiently. Data integration techniques such as Extract, Transform, Load (ETL) processes, APIs, and middleware solutions are employed to facilitate this seamless flow of data across systems.
- **ETL Processes:** Extract, Transform, Load processes are used to integrate data from various sources into a unified system.
- **API Management:** Using API gateways and management tools like AWS API Gateway, Postman, and Swagger to handle data exchange between components.
- **Real-Time Data Processing:** Apache Kafka or AWS Kinesis for processing streaming data in real-time.

4.4.2 Data Analytics

Data analytics involves interpreting the integrated data to extract meaningful insights, which can drive informed decision-making. Advanced analytics tools and methodologies are utilized to analyze the data, revealing underlying relationships and predicting future trends.

- **Advanced Analytics Tools:** The framework leverages advanced analytics tools and methodologies to interpret the integrated data effectively. These tools enable the extraction of meaningful patterns and trends from the data, providing insights that can drive informed decision-making. Machine learning algorithms, statistical models, and data visualization techniques are among the tools utilized to analyze the data.
- **Predictive Modeling for Resource Optimization:** One of the key objectives of incorporating data integration and analytics into the framework is the development of predictive models that can optimize campus resources and services. By analyzing patterns in data related to campus operations, energy consumption, space utilization, and student engagement, predictive models can forecast future demands and trends. This enables proactive management of resources, improving efficiency and reducing costs, while enhancing the overall campus experience.
- **Facilitating Informed Decision-Making:** The integration of comprehensive data analytics within the framework empowers stakeholders to make informed decisions. By providing access to real-time insights and predictive analytics, administrators, faculty, and staff can optimize operations, tailor services to meet the needs of the campus community, and respond swiftly to emerging challenges.

4.4.3 Data Security

Ensuring the security and privacy of data is fundamental in the smart campus environment. This subsection focuses on the strategies and technologies employed to protect data throughout its lifecycle, from collection to processing and storage.

- **Data Encryption:** All data, whether at rest or in transit, is encrypted using robust encryption standards. This includes the use of HTTPS to encrypt data transmitted over the network, ensuring that data in transit between clients and servers is protected against interception and tampering. This ensures that even if data is intercepted or accessed by unauthorized entities, it remains unintelligible and secure.
- **Access Controls:** Implementing strict access controls to ensure that only authorized personnel can access sensitive data. Role-based access control (RBAC) and multi-factor authentication (MFA) are utilized to enhance security.
- **Data Governance Policies:** Establishing comprehensive data governance policies that define how data is to be handled, stored, and processed. This includes data quality checks, validation processes, and regular audits to ensure compliance with security standards.

- **Monitoring and Incident Response:** Continuous monitoring of data systems for potential security breaches and anomalies. Incident response protocols are established to swiftly address and mitigate any security incidents.
- **Authentication and Authorization:** Utilizing authentication and authorization methods based on JWT (JSON Web Tokens) or OAuth to ensure secure interactions for users.
- **Server-Side Security:** Implementing security measures on the server-side, such as secure server-side frameworks like Node.js, Express, and Next.js, to build scalable and secure applications.

By integrating data from various sources, employing advanced analytics, and ensuring robust data security, the Data Component of the framework enhances the ability to derive actionable insights and maintain the integrity and trustworthiness of the data. The subsequent sections will continue to elaborate on the comprehensive framework for developing web applications in smart campus environments.

4.5 Agile Methodology

Central to this architecture, the application component integrates these layers into a fluid, agile methodology's iterative process, highlighting the framework's adaptability and commitment to continuous refinement. This process not only facilitates application development but also aligns with evolving technological and user requirements, maintaining the applications' relevance and effectiveness. Core practices include:

- **Scrum:** For structured development cycles with defined roles and time-boxed sprints.
- **Kanban**: For visualizing workflow and managing work-in-progress to enhance efficiency.
- **Continuous Integration/Continuous Deployment (CI/CD):** Using tools like GitHub Actions or Jenkins to automate testing and deployment processes.

Together, these components forge a robust framework designed for the dynamic, interconnected nature of smart campuses. This integrated approach not only addresses the technical complexities in smart campus ecosystems but also prioritizes a seamless, user-focused experience, underpinning the framework's agility and continuous evolution in response to the changing landscape of campus life and technology.

4.6 Front-end Technologies

4.6.1 React

React was selected as a primary technology for building the user interface. We will focus into the reasons behind choosing React and how it contributed to the development process.

- **Component-Based Architecture:** React's core strength lies in its component-based architecture. This approach allows developers to build encapsulated components that manage their state, leading to more manageable and reusable code [97]. For applications, this meant that UI components such as cards, user profiles, and scheduling interfaces could be developed independently and reused, enhancing development efficiency.

- **Declarative UI:** React makes it straightforward to create interactive UIs. By designing simple views for each state in the application, React efficiently updates and renders the right components when data changes [98]. This declarative nature simplifies the code and makes it more predictable, a key advantage in developing the user interfaces for applications

- **Performance:** React implements a virtual DOM, an in-memory representation of the real DOM, which allows for optimal updating of the web application's UI [99]. This results in efficient performance, crucial for real-time responsiveness and smooth user interactions.

- **Strong Community and Ecosystem:** React's widespread adoption and robust community mean extensive resources, libraries, and tools are available [100]. This ecosystem was invaluable in addressing various developmental challenges, providing well-tested libraries and frameworks that could be integrated into projects.

4.6.2 Typescript

Following the decision to utilize React for the frontend development, TypeScript was chosen as the primary programming language. We will explore the rationale behind selecting TypeScript and its impact on the development process.

- **Enhanced Code Quality and Maintainability:** TypeScript, as a statically typed superset of JavaScript, offers advantages in terms of code quality and maintainability. Static typing helps in detecting errors early in the development process, which is crucial for ensuring the robustness of applications [101].

- **Improved Developer Productivity:** TypeScript's powerful type system, including interfaces and generics, enhances developer productivity and collaboration. It provides better tooling support with autocompletion, type checking, and refactoring capabilities, which streamline the development process [102].

- **Scalability:** The applications requirements for scalability in a complex and evolving smart campus ecosystem made TypeScript an ideal choice. Its ability to handle large codebases and ensure code consistency across the development team is highly beneficial.

- **Compatibility with React:** TypeScript's compatibility with React significantly streamlines the development process. It allows for writing safer and more predictable components, enhancing the overall stability and reliability of the application [103].

4.6.3 Next.js

Next.js [104] is an open-source React full-stack development web framework that enhances the capabilities of React applications through features like server-side rendering and static site generation. Known for its efficiency and ease of use, Next.js is a popular choice for developers building modern web applications. We dive into the core features of this framework in the Table 4.1.

4.7 Back-end Technologies

4.7.1 tRPC

tRPC [105] is a framework designed for building and consuming fully typesafe APIs in TypeScript without relying on schemas or code generation. It is particularly beneficial for full-stack TypeScript developers, providing a streamlined process for creating APIs that are typesafe and efficient. tRPC's main advantage lies in its ability to unify backend and frontend type definitions, ensuring type safety across the entire stack. This is accomplished by linking the backend typing to the client typing via an internal routing component [106], which allows for real-time validation of code during creation, a significant shift from traditional compiling or code generation stages.

In tRPC, the first step in implementation is defining a router, which exposes endpoints to the frontend and enables type safety. Procedures within tRPC function similarly to REST endpoints, representing the operations for data queries and mutations. tRPC also supports the creation of separate routers for different sets of functions, which can be merged into a single router core, allowing for complex data retrieval and manipulation.

Table 4.1: Core Features of Next.js

Feature Category	**Details**
Server-Side Rendering (SSR)	Enables rendering pages on the server, which improves performance, SEO, and user experience.
Static Site Generation (SSG)	Supports pre-rendering pages at build time, ideal for pages with infrequent updates.
Incremental Static Regeneration (ISR)	Combines SSG benefits with on-demand rendering, allowing post-deployment content updates.
API Routes	Facilitates creation of API endpoints within the application, aiding in full-stack development.
CSS and Sass Support	Offers built-in support for CSS and Sass, including compatibility with CSS-in-JS libraries.
Fast Refresh	Provides instant feedback on React component edits, enhancing the developer experience.
File-system Routing	Utilizes a file-system-based routing mechanism, with automatic routing based on ‘pages‘ directory.
Image Optimization	The Image component automatically optimizes images.
TypeScript Support	Includes out-of-the-box support for TypeScript, enabling type-checking benefits.

One of the key benefits of tRPC is its framework agnosticism, making it adaptable to various backend frameworks. This flexibility is achieved through API Handlers or adapters that convert requests from different frameworks into a format that tRPC can process. Furthermore, tRPC is relatively lightweight and does not require extensive setup like some other frameworks, such as GraphQL, making it an attractive option for projects where typescript-centric development are suitable.

Overall, tRPC offers a compelling solution for TypeScript developers looking for a simple yet powerful tool to ensure type safety and efficient API development, especially in scenarios where both client and server implementations are part of the same project.

4.7.2 MySQL

In the proposed framework for web applications development in smart campus environments, MySQL was selected as the database management system. This subsection discusses the reasons behind choosing MySQL and its relevance in the framework's architecture.

- **Reliability and Maturity:** MySQL is renowned for its reliability and maturity as a database solution. Its stability and robustness are essential in managing complex data relationships and sensitive information within a smart campus environment.
- **Scalability and Performance:** MySQL's scalability ensures that the database can grow in line with the increasing demands of a smart campus application. It offers high-performance capabilities, vital for handling multiple transactions and user interactions efficiently.
- **Flexibility and Compatibility:** The flexibility of MySQL, especially its compatibility with various programming languages and platforms, makes it a versatile choice. This compatibility is crucial for seamless integration with diverse technologies typically used in smart campus applications.
- **Community Support and Resources:** The extensive community support and documentation available for MySQL provide invaluable resources for development, troubleshooting, and optimization, aiding in the efficient implementation of database solutions in a smart campus context.

The selection of MySQL as the database management system is a fundamental aspect of the proposed framework for web applications in smart campus environments. Its reliability, scalability, flexibility, and extensive community support make it an excellent choice for robust data management in such settings. These characteristics of MySQL align well with the objectives of creating efficient, scalable, and secure web applications for smart campuses.

4.7.3 PrismaORM

Prisma [107] is an open-source Object Relational Mapping (ORM) tool that simplifies database operations for developers by providing an easy-to-use API to interact with databases. It supports various database engines, including PostgreSQL, MySQL, SQLite, SQL Server, and MongoDB, among others. Prisma is known for its focus on developer productivity, safety, and ease of use.

Prisma offers the generation of a type-safe client, derived from the schema definitions. This approach guarantees the validation of database queries at the compile stage, significantly diminishing the potential for runtime errors and elevating the overall quality of the code. Additionally, Prisma introduces a sophisticated query engine capable of converting high-level operations into optimized database queries. This functionality enables developers to execute complex queries with relative ease, addressing a critical need within the development community.

Prisma also simplifies database migration and management through Prisma Migrate, which automates the creation and execution of database migrations following schema alterations. This method provides a structured and declarative strategy for database version control and modification application.

Prisma's adaptability is evident in its support for multiple database systems, ensuring its applicability across a diverse range of projects and technological requirements. Additionally, it offers introspection capabilities, allowing for the seamless integration of Prisma into existing databases and facilitating its adoption in projects with legacy systems.

The advantages of utilizing Prisma are multiple. It significantly boosts developer productivity by simplifying database operations and reducing development timeframes. The type-safe nature of the Prisma Client minimizes errors commonly associated with database interactions, while the framework's intuitive API and extensive documentation make it accessible to developers of varying skill levels. Lastly, Prisma's efficient query engine and multi-database support render it an ideal choice for applications at any scale, from nascent startups to expansive enterprise systems.

4.8 Additional tools

4.8.1 Git & GitHub

Git and GitHub are foundational tools in modern software development. Git, a distributed version control system, enables developers to track changes in code efficiently. GitHub, meanwhile, extends Git's capabilities with web-based hosting and collaboration features. For a comprehensive understanding, refer to Chacon and Straub's "Pro Git" book for Git [108] and the official GitHub Documentation [109].

Table 4.2: Core Features of Git

Feature	**Details**
Distributed Version Control	Unlike centralized version control systems, Git gives every developer a local copy of the entire development history, enhancing speed and flexibility.
Branching and Merging	Git's branching model allows multiple developers to work on different features simultaneously without interfering with each other. Merging brings these branches together into a single unified history.
Data Integrity	Git is designed to ensure the integrity of source code. Each file and commit is checksummed, and the repository's history is stored such that it's impossible to change without being detected.

Table 4.3: Core Features of GitHub

Feature	**Details**
Repository Hosting	GitHub hosts millions of Git repositories, making it easy for teams to store and manage their codebases.
Pull Requests and Code Review	GitHub's pull request system streamlines code review. Developers can discuss changes, request improvements, and push follow-up commits before changes are merged into the main branch.
Issue Tracking	GitHub provides issue tracking tools that allow teams to organize tasks, enhancements, and bugs associated with their projects.
GitHub Actions	This CI/CD feature automates workflows, allowing automatic build, test, and deployment processes based on repository events.

GitHub stands as a important tool in the area of software development, offering integration with a wide range of development tools. It is compatible with various Integrated

Development Environments (IDEs), project management tools, and Continuous Integration/Continuous Deployment (CI/CD) pipelines. This integration significantly enhances the development workflow process for developers. Furthermore, GitHub plays a vital role in the open-source community. It serves as a central platform for open-source projects, creating an active community where developers from around the globe can easily contribute to public repositories. This fosters collaboration and innovation in the software development field.

In addition to its integration capabilities, GitHub places a strong emphasis on security and access control. The platform provides robust access control mechanisms for repositories, empowering repository owners to meticulously manage who has the ability to read and contribute to the codebase. This feature is crucial for maintaining the integrity and confidentiality of the code. Additionally, GitHub is equipped with various security features designed to safeguard the code. These features include security advisories and automated vulnerability scanning, which play a critical role in identifying and addressing potential security threats. This comprehensive approach to security ensures that the code hosted on GitHub remains secure and reliable.

To sum, Git and GitHub provides robust tools for version control and collaboration as seen in Table 4.2 and 4.3. Git's distributed nature and powerful branching capabilities, combined with GitHub's hosting, collaboration features, and community ecosystem, make them indispensable for both individual developers and large teams.

4.8.2 v0

v0 by Vercel is an innovative generative UI tool that has been gaining attention in the web development community. As of November 2023, it's in private beta, but access is available through a waitlist. The tool is designed to streamline the UI generation process using artificial intelligence. A comprehensive review of v0 based on the available information in Vercel's announcement [110] describe some features like:

- **Generative UI Creation:** v0 allows you to quickly generate UIs by simply typing in text prompts. For example, entering ”user profile” generates several UI options based on that prompt.
- **Integration with React and Tailwind CSS:** The tool generates code that integrates well with React and Tailwind CSS, making it easy to incorporate into existing projects.
- **Shadcn UI Compatibility:** v0 assumes the use of Shadcn UI, a collection of reusable components, enhancing its usability and customizability in projects.
- **Iterative Development** You can iterate on the initial UIs using prompts, creating skeletons that are easy to modify and extend.

4.8.3 Vercel

Vercel is a cloud platform for web applications that provides developers with the tools needed to build, deploy, and scale modern web applications. It integrates with frontend frameworks like React, Next.js, and others, offering a robust and efficient environment for deploying web applications. Some key features that made us to choose Vercel as our hosting provider are:

- **Automatic Deployments:** Vercel provides a deployment process where every push to the repository triggers an automatic build and deployment, ensuring the availability of the latest version of the application.
- **Serverless Functions:** Allows developers to write server-side code that executes without managing server infrastructure. These functions can be used for handling API requests, form submissions, and other backend logic.
- **Integration with Next.js:** As the creators of Next.js, Vercel offers optimized support for this framework, including features like static site generation (SSG) and server-side rendering (SSR), making it an ideal choice for Next.js applications.
- **Analytics and Monitoring:** Vercel provides built-in analytics to monitor the performance and usage of applications, offering insights that can help optimize the user experience.

4.8.4 NextAuth

NextAuth.js is a complete open-source authentication solution for Next.js applications. It provides a robust and flexible way to add authentication to web applications, supporting various authentication methods including OAuth, Email/Password, and custom authentication strategies. Using NextAuth.js provides important benefits to our framework like:

- **Easy Integration with Next.js:** Designed specifically for Next.js, NextAuth.js allows and easy setup and configuration in Next.js applications.
- **Multiple Authentication Providers:** Supports a wide range of providers including Google, Facebook, GitHub, and more. This flexibility makes it easy to implement different authentication methods based on the application's needs.
- **Security:** Implements secure practices by default, including secure cookies and token encryption, ensuring that user data and authentication tokens are handled safely.
- **Customization:** Highly customizable, allowing developers to define custom authentication flows, adapt session handling, and configure callbacks for additional control over the authentication process.

- **Serverless Compatibility:** Works with serverless architectures, making it a great fit for modern web applications that leverage serverless functions for backend logic.

Chapter 5

Results and Discussion

This chapter presents the results of the thesis work through the implementation of a proof-of-concept application, showcasing the practical application of the proposed web application development framework within a smart campus environment. The chapter begins with an overview of the requirements gathering process. Following this, the chapter details the implementation of the application in alignment with the proposed web application development framework and its evaluation metrics. The chapter concludes with an evaluation of how the proof-of-concept aligns with the Smart Campus Framework, demonstrating the framework's effectiveness in enhancing operational efficiency, sustainability, and user experience. The findings underscore the potential impact and scalability of the proposed framework, highlighting its capacity to drive innovation and improve campus operations and user satisfaction.

5.1 Requirements Gathering

This section presents the process of identifying key requirements for a web application designed to facilitate peer-tutoring on campus. Our focus was on understanding the dual role of students as both learners and tutors. We emphasized the functional requirements essential for user registration, session management, and feedback mechanisms, ensuring these align with the proposed database schema. Additionally, non-functional requirements like usability, security, and scalability were considered vital for a seamless user experience. This foundational step was crucial in shaping the application to effectively meet the dynamic needs of a smart campus environment, while offering a user-friendly interface for an efficient knowledge exchange between students.

5.1.1 Stakeholder Analysis

In the context of developing a peer-tutoring web application, a comprehensive stakeholder analysis is critical for ensuring that the platform meets the diverse needs of its user base. In table 5.1 we explore the various stakeholders involved in the peer-tutoring ecosystem and their respective requirements and expectations.

Table 5.1: Stakeholder Analysis for Peer-Tutoring Web Application

Stakeholder	**Description**
Students	The primary users of the application, students play dual roles as tutors and tutees. Their needs include easy navigation for registering as a tutor or tutee, selecting subjects, scheduling sessions, and providing feedback. The platform must be intuitive and accommodating to their academic and scheduling needs.
University Administration	As overseers of the campus's academic and extracurricular activities, their interest lies in the application's ability to enhance the educational experience. They require robust data reporting tools for tracking usage patterns and student performance.
IT Department	Responsible for the technical implementation and maintenance of the application. They are concerned with the scalability, security, and integration of the platform with existing campus systems.
Academic Departments	Interested in how the application can support their curriculum and assist students struggling with their courses. They might require access to data that helps them understand common tutoring needs in their subjects.

This analysis led to a clear understanding of the varied requirements and constraints each stakeholder group brought to the project. It enabled the development of a well-rounded application that not only concerns to the academic and scheduling flexibility required by students but also aligns with the administrative, technical, and security standards of the university.

5.1.2 Functional Requirements

The functional requirements for the peer-tutoring web application are important in defining what the system is expected to do. These requirements are specifically designed to solve to the needs of both tutors and tutees within the smart campus environment. The following points detail these key requirements in Table 5.2.

Table 5.2: Functional Requirements for Peer-Tutoring Web Application

Requirement	**Description**
User Registration and Role Management	Users can register and create profiles with academic details. Automatic assignment of 'student' role and option to sign up as tutors.
Subject and Tutor Selection	Feature for students to select subjects and match with available tutors based on preferences.
Scheduling and Session Management	Students can schedule sessions specifying details; tutors view and manage requests. System tracks session status.
Feedback and Rating System	Post-session, both parties can rate and comment. System records this for quality control and performance tracking.
Notification and Communication	Notification system for session updates and direct communication feature within the platform.
Data Integration and Reporting	Integration with campus database and reporting tools for tracking usage and session statistics.

These functional requirements are essential to ensure that the web application is user-friendly, efficient, and meets the specific needs of the campus community for peer-tutoring services. They form the backbone of the system's operations and user interactions, ensuring a seamless and productive tutoring experience.

5.1.3 Non-functional requirements

Ensuring the usability, reliability, and effectiveness of the peer-tutoring web application is part of the Non-functional requirements. These requirements don't directly impact the specific activities of the system but rather define the system's operational qualities and constraints. In Table 5.3 are the key non-functional requirements identified for this application:

Table 5.3: Non-Functional Requirements for Peer-Tutoring Web Application

Requirement	**Description**
Usability	Application should be user-friendly and accessible, with an intuitive interface suitable for all users.
Performance	Capable of handling multiple requests simultaneously with optimized load times for efficient user experience.
Scalability	Must be scalable to accommodate growing user numbers and data, maintaining functionality under varied loads.
Security	Robust security measures for data protection, including compliance with data protection regulations and campus policies.
Reliability	Minimal downtime, quick recovery from failures, regular backups, and a robust disaster recovery plan.
Maintainability	Designed for easy maintenance and updates, with clear documentation and modular design for ongoing support.
Integration	Seamless integration with existing campus systems and databases, ensuring compatibility with various platforms and devices.

These non-functional requirements play an important role in ensuring that the peer-tutoring web application not only meets the immediate needs of its users but also remains a reliable, secure, and efficient tool in the long term. They form the foundation for a sustainable and adaptable application that aligns with the dynamic environment of a smart campus.

5.2 Design and Development

This section analyzes the intricate processes involved in the design and development of the peer-tutoring web application, tailored for the smart campus environment. This phase translates the gathered requirements into a tangible and functional system. The section is structured to provide a comprehensive insight into the journey from conceptualization to realization of the peer-tutoring web application, highlighting the technical strategies and design decisions that contributed to its successful deployment in a smart campus setting.

5.2.1 Use Case Diagram

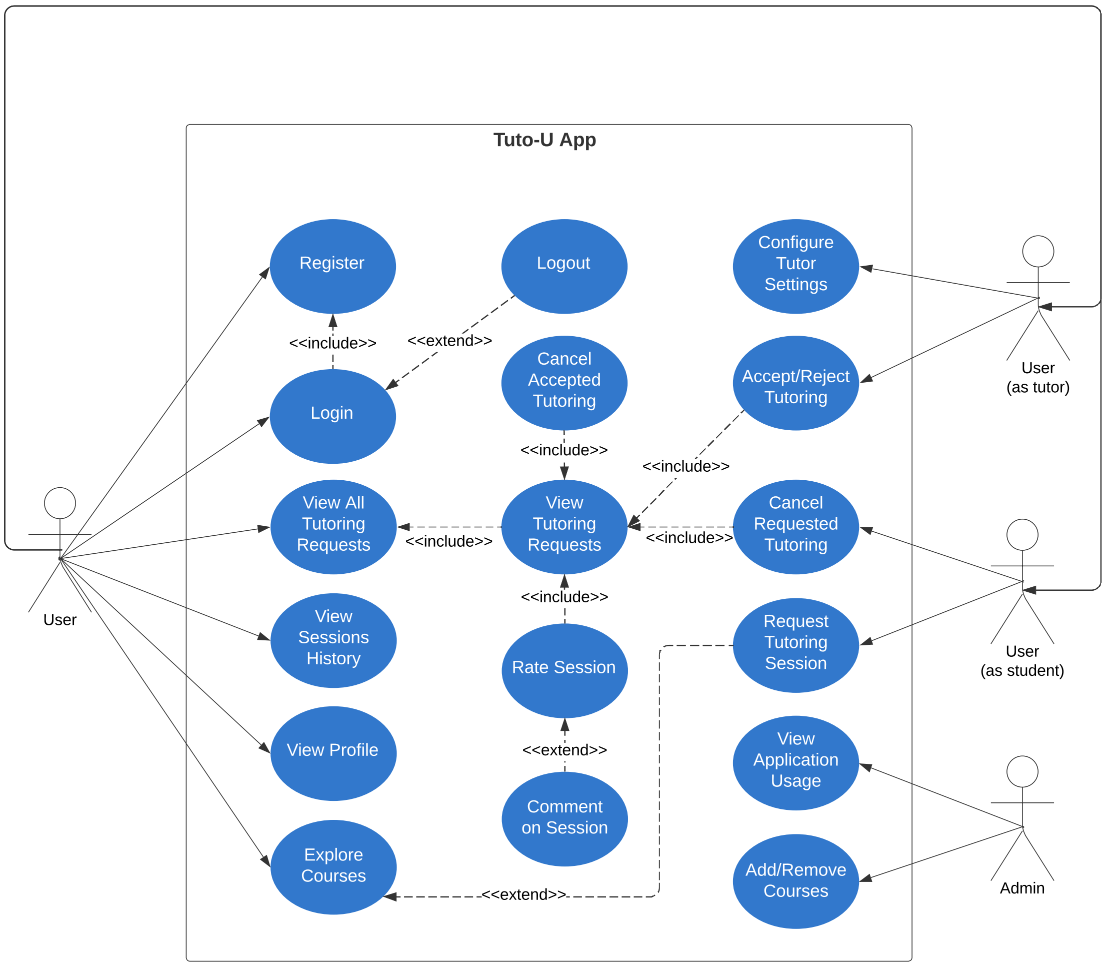


Figure 5.1: Proof-of-concept use case diagram.

The use case diagram is an integral part of the system design process, providing a graphical representation of the interactions between users (actors) and the system (the peer-tutoring web application). Fig 5.1 illustrates the various functionalities available to different user roles and how these roles interact with the system. The key components of the use case diagram are:

1. **Actors:**
   - **Student:** Can register, log in, browse tutors, request sessions, provide feedback to tutors.

- **Tutor:** Inherits Student functionalities and can additionally offer tutoring, manage session requests, and provide feedback for students.
- **Administrator:** Manages the system, oversees user activity, and accesses reports.

2. **Use Cases:**

   - **User Registration and Login:** Users can register and subsequently log in to access the system.
   - **Browse Tutors and Request Sessions:** Students can browse available tutors and request tutoring sessions.
   - **Offer Tutoring Services:** Tutors can configure their settings like select which subjects they can be requested, their availability time or pricing information.
   - **Manage Session Requests:** Tutors can accept, reject, or propose changes to session requests. Students can cancel the session if necessary.
   - **Feedback Provision:** After a session, both tutors and students can provide ratings and feedback.
   - **View Tutoring Sessions History:** Allows a user to see a history of all tutoring sessions they have participated in.
   - **System Management and Reporting:** Administrators can manage the system and access comprehensive reports.

3. **Relationships and Interactions:**

   - The diagram illustrates interactions between actors and use cases, such as how a Student requests a session and how a Tutor responds to it.
   - It also shows administrative interactions like view system usage and logs.

5.2.2 System Architecture

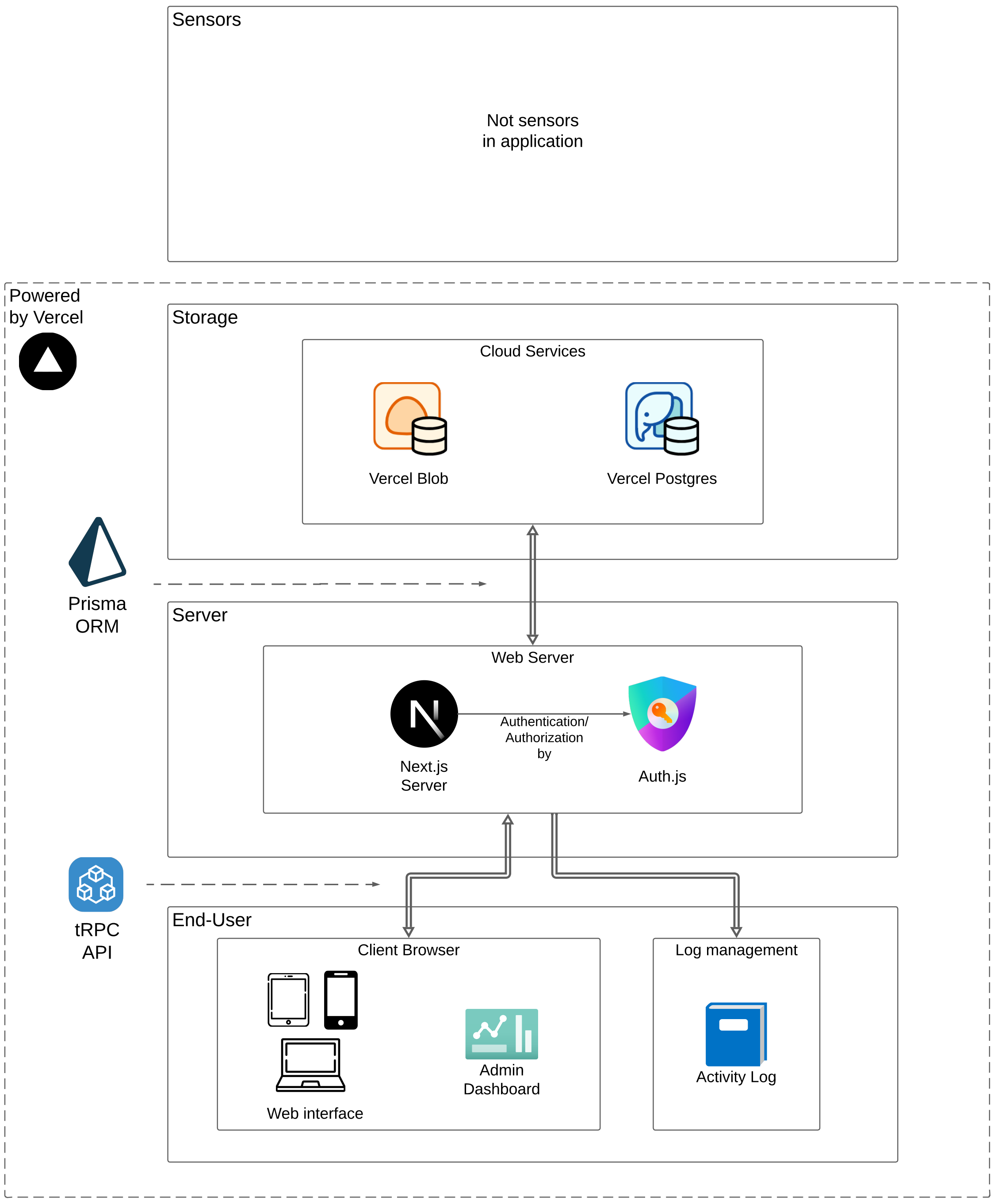


Figure 5.2: Proof-of-concept application architecture.

The system architecture (Figure 5.2) of the peer-tutoring app is illustrated across the four integral layers of the application component from proposed framework, which collectively contribute to a robust and efficient application ecosystem. This architecture, as seen in our framework, ensures a seamless flow from data acquisition to user interaction, although the Sensors layer in our context remains unutilized.

- **Sensors Layer:** The first layer of the architecture, the 'Sensors Layer', is conceptual within the application. In traditional systems, this layer is bustling with data-gathering components; however, application's model currently does not incorporate physical data collection through sensors. Despite this, the inclusion of this layer in our structural blueprint allows for scalability and future integration of data-collection mechanisms if required.
- **Storage Layer:** Ascending to the 'Storage Layer', we integrate Vercel Blob Storage, utilized for managing the vast array of images and media that enrich the user interface. For structured storage, a Vercel Postgres database was created, employed as the primary database system, ensuring data integrity, security, and accessibility. Prisma ORM acts as the bridge between Next.js and the database, providing a type-safe ORM (Object-Relational Mapping) that simplifies the interaction with the database with high efficiency.
- **Server Layer:** In the 'Server Layer', our server-side operations are powered by Next.js, which brings server-side rendering and static generation capabilities to our React application. This framework enhances performance and improves SEO by serving pre-rendered pages to the browser. TypeScript is the chosen language to ensure type safety and improve the development experience with robust typing. This layer is optimized for performance through strategic caching, which drastically reduces load times and server response delays. Additionally, NextAuth manages authentication and authorization, providing secure access control and user management in the application.
- **End-User Layer:** The final tier is the 'End-User Layer', where the rich web-based interface crafted in React creates an engaging and interactive experience for the users. React's component-based architecture, paired with the power of Next.js for server rendering, allows for a highly responsive and dynamic user interface. User interface captures can be seen in Appendix A.

The application is deployed and managed on Vercel, which provides hosting, serverless functions, and other cloud services. The general application flow can be described as follows:

- **Client Interaction:** Users interact with the web interface via their browsers, sending requests to the server.

- **Server Processing:** The Next.js server processes these requests, using NextAuth for authentication and Prisma ORM for database operations.
- **Storage Access:** The server interacts with cloud storage services (Vercel Blob and Vercel Postgres) to fetch or store data as needed.
- **Data Transfer:** The tRPC API facilitates communication between the client-side and server-side, ensuring data is transferred securely and efficiently.
- **Activity Logging:** User activities are logged for monitoring and security purposes via Vercel dashboard.

5.2.3 Database Schema

The database schema for our smart campus web application plays an important role in managing and organizing the data effectively. The schema is designed to handle the complexities of a dynamic educational environment, ensuring data integrity, accessibility, and scalability. This section provides an in-depth look at the database schema, illustrated by the Entity-Relationship (ER) diagram in Appendix B. The ER diagram visually represents the relationships between these tables, highlighting the connections and dependencies crucial for maintaining the integrity and functionality of the application. Each relationship in the diagram is designed to ensure integrity and support the application's requirements for data retrieval and manipulation. The schema also supports scalability and performance optimization. The use of indexing, foreign keys, and normalization principles ensures efficient data management and quick access to relevant information, essential for a responsive user experience.

5.3 Alignment with the Proposed Web Application Development Framework

The proof-of-concept application closely follows the framework's principles, showcasing several key aspects:

5.3.1 User-Centered Design

The application employs user-centered design principles to ensure accessibility and a seamless user experience. This is reflected in the intuitive interface, which allows users to navigate the system easily, register as tutors or students, schedule sessions, and provide feedback. The design process involved usability testing and participatory design practices, engaging real users to gather insights and iteratively refine the interface.

5.3.2 Scalable Architecture

The application's architecture is designed to accommodate the growing needs of a smart campus. By working with modern web technologies like Next.js, Prisma ORM, and serverless functions provided by Vercel, the system ensures scalability and performance optimization. The architecture includes multiple layers such as the storage, server, and end-user layers, each contributing to the overall robustness and efficiency of the application.

5.3.3 Data Integration and Analytics

The application integrates various data sources and employs advanced analytics to provide meaningful insights. This integration is facilitated through APIs that enable the retrieval and processing of data, supporting informed decision-making processes. The framework's emphasis on data-driven approaches is evident in features like real-time session tracking, user feedback analysis, and performance metrics for tutors and students.

5.3.4 Security and Privacy

Ensuring data security and privacy is a critical component of the framework, and the application implements robust measures to protect user information. Authentication and authorization are managed using NextAuth, ensuring secure access control. Additionally, the system complies with data protection regulations, incorporating encryption and regular security audits to safeguard sensitive data.

5.3.5 Responsive and Agile Methodologies

The development process followed agile methodologies, allowing for flexibility and rapid iterations. This approach enabled the team to respond to user feedback promptly and make necessary adjustments, ensuring that the application remained aligned with user needs and technological advancements. The use of modern development tools and practices, such as continuous integration and deployment (CI/CD) pipelines provided bu Vercel ecosystem, further streamlined the development workflow.

5.4 Application Metrics

In this section, we will discuss the metrics used to evaluate the performance, accessibility, and best practices of web applications using Lighthouse, a popular open-source tool developed by Google. Lighthouse provides an in-depth analysis and scoring system across several categories, including Performance, Accessibility, and Best Practices. The scores range from 0 to 100, with higher scores indicating better performance.

5.4.1 Performance Metrics

Performance metrics assess how quickly a web page loads and becomes interactive. Lighthouse measures several key performance indicators:

- **First Contentful Paint (FCP):** This metric measures the time it takes for the first piece of content to be rendered on the screen. It is crucial as it gives users the first visual feedback that the page is loading.
- **Largest Contentful Paint (LCP):** LCP measures the time it takes for the largest piece of content (e.g., an image or a large block of text) to be rendered on the screen. It is a significant indicator of page load performance.
- **Total Blocking Time (TBT):** This metric evaluates the time during which the main thread is blocked and unable to respond to user input. It is essential for assessing how responsive the application is.
- **Cumulative Layout Shift (CLS):** CLS measures the visual stability of the page by tracking unexpected layout shifts during the page's lifecycle. A low CLS score indicates a more stable and visually appealing user experience.
- **Speed Index (CI):** This metric measures how quickly the content of a page is visibly populated. It reflects the user's experience of how fast the page loads.

Lighthouse aggregates these metrics to calculate an overall performance score. This score is weighted based on the importance of each metric, reflecting a comprehensive assessment of the application's performance. The score weights are detailed in Table 5.4. The complete methodology for each score calculation can be found in the Lighthouse documentation [111].

Table 5.4: Key performance indicators weight for overall performance score calculation

Performance Indicator	**Weight**
First Contentful Paint	10%
Largest Contentful Paint	25%
Total Blocking Time	30%
Cumulative Layout Shift	25%
Speed Index	10%

5.4.2 Accessibility Metrics

Accessibility metrics evaluate how accessible a web application is to users with disabilities. This includes the use of proper HTML elements, ARIA attributes, and overall adherence to accessibility standards. Key areas evaluated by Lighthouse include:

- **Semantic HTML:** Proper use of HTML elements to ensure that the document structure is logical and meaningful.
- **ARIA Roles:** Appropriate use of ARIA (Accessible Rich Internet Applications) roles to enhance the accessibility of dynamic content.
- **Color Contrast:** Ensuring sufficient contrast between text and background colors to improve readability for users with visual impairments.
- **Keyboard Navigation:** Ensuring that all interactive elements are accessible via keyboard navigation.

Lighthouse calculates the accessibility score by checking each element of the page against established guidelines (e.g., WCAG). The score is an aggregation of the individual checks, weighted by their impact on accessibility. The 23 evaluated indicators are shown in Table 5.5.

Table 5.5: Accessibility audits from Lighthouse evaluation

Accessibility Indicators
[aria-*] attributes match their roles
[aria-hidden="true"] is not present on the document <body>
[role]s have all required [aria-*] attributes
[aria-*] attributes have valid values
[aria-*] attributes are valid and not misspelled
Buttons have an accessible name
Image elements have [alt] attributes
[user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.
ARIA attributes are used as specified for the element's role
Elements use only permitted ARIA attributes
[role] values are valid
Background and foreground colors have a sufficient contrast ratio
Document has a <title> element
<html> element has a [lang] attribute
<html> element has a valid value for its [lang] attribute
Links have a discernible name
Lists contain only <li> elements and script supporting elements (<script> and <template>).
No element has a [tabindex] value greater than 0
Touch targets have sufficient size and spacing.
Heading elements appear in a sequentially-descending order
Values assigned to role="" are valid ARIA roles.
Deprecated ARIA roles were not used
Image elements do not have [alt] attributes that are redundant text.

5.4.3 Best Practices Metrics

Best practices metrics evaluate adherence to web development best practices. This includes security, performance, and progressive web app (PWA) compliance. Key areas evaluated by Lighthouse include:

- **HTTPS:** Ensuring that the application is served over HTTPS to protect data integrity and privacy.
- **No Vulnerable Libraries:** Checking that the application does not use libraries with known security vulnerabilities.
- **Optimized Images:** Ensuring that images are appropriately sized and optimized for faster load times.
- **Efficient Cache Policy:** Implementing a proper cache policy to enhance performance and reduce load times.

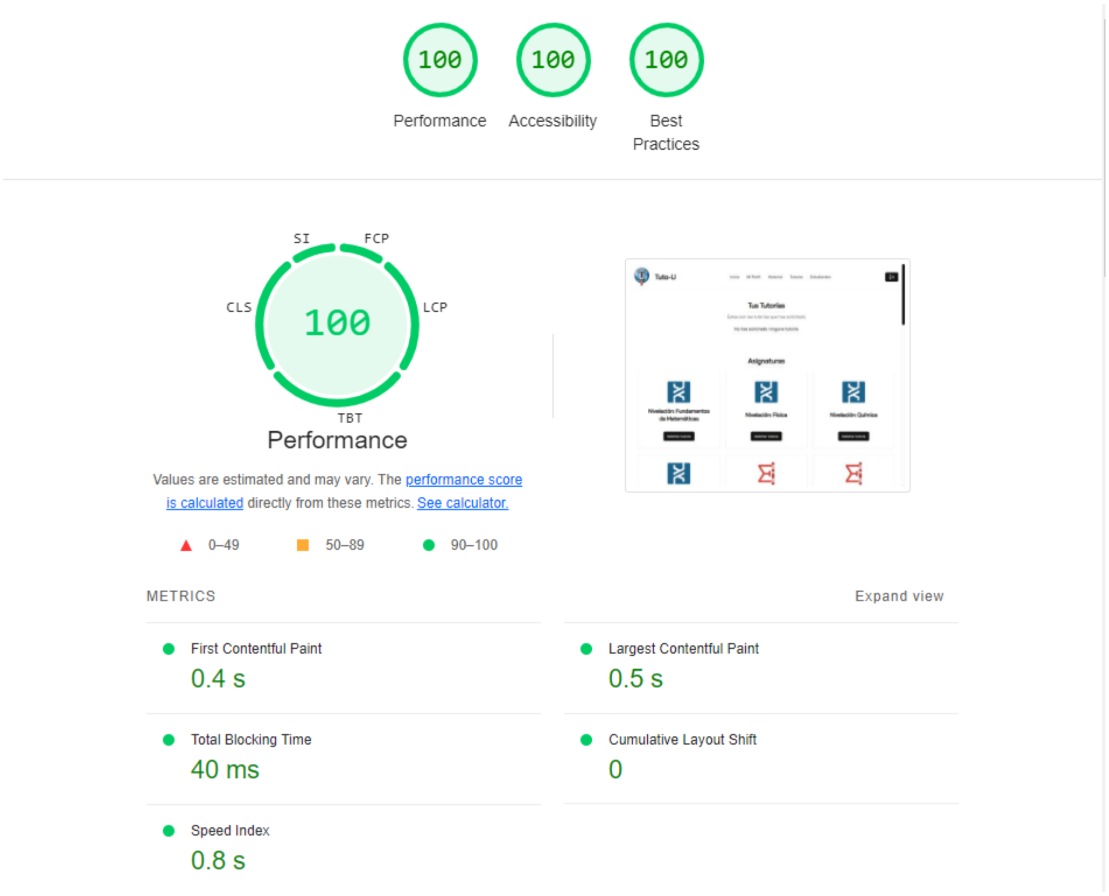
The best practices score is calculated based on the adherence to a set of predefined guidelines and recommendations. Each guideline is weighted, and the overall score is a reflection of the application's compliance with these best practices. The 14 evaluated indicators are shown in Table 5.6.

Table 5.6: Best Practices audits from Lighthouse evaluation

Best Practices Indicators
Uses HTTPS
Avoids deprecated APIs
Avoids third-party cookies
Allows users to paste into input fields
Avoids requesting the geolocation permission on page load
Avoids requesting the notification permission on page load
Displays images with correct aspect ratio
Serves images with appropriate resolution
Has a <meta name="viewport"> tag with width or initial-scale
Page has the HTML doctype
Properly defines charset
No browser errors logged to the console
No issues in the Issues panel in Chrome Devtools
Page has valid source maps

5.4.4 Comparison of Application Versions Lighthouse Results

The provided screenshots in Figures 5.3 and 5.4 illustrate the results of Lighthouse audits for two different versions of the web application, showcasing significant advancements and optimizations in the current version.



100 Performance

100 Accessibility

100 Best Practices

SI FCP CLS LCP TBT

100

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint
0.4 s

● Largest Contentful Paint
0.5 s

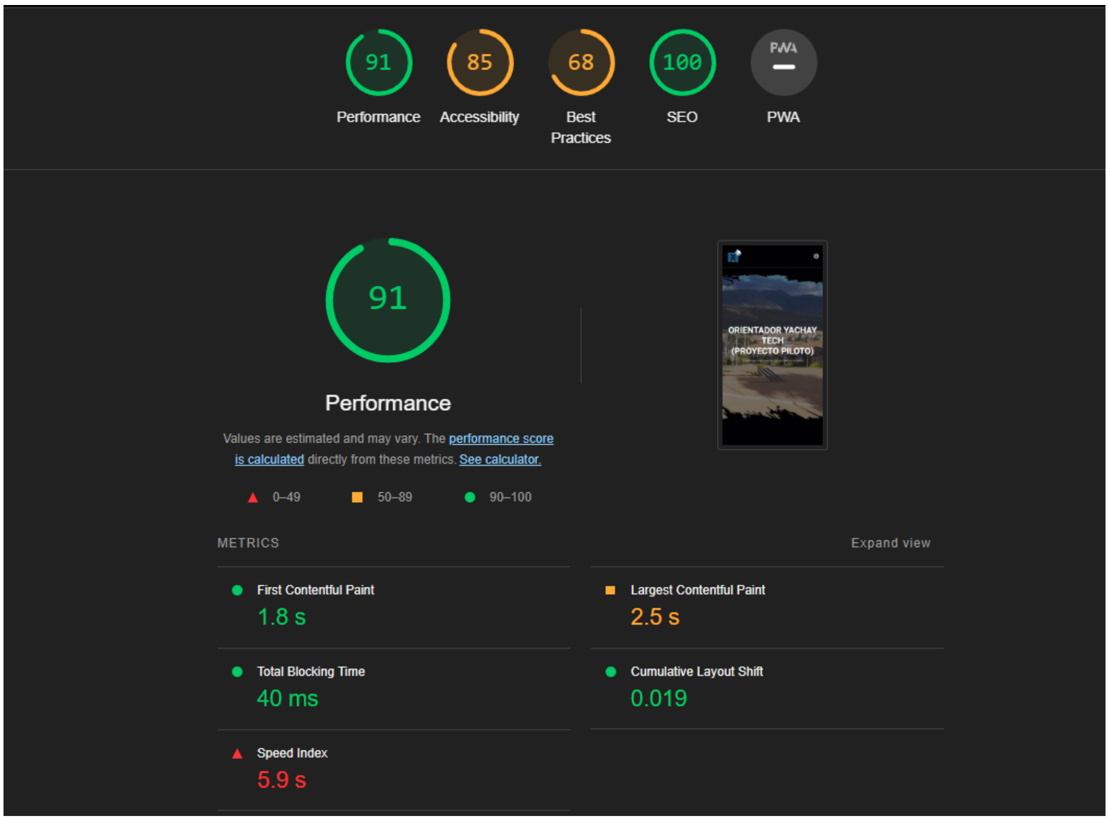
● Total Blocking Time
40 ms

● Cumulative Layout Shift
0

● Speed Index
0.8 s

Figure 5.3: Lighthouse results of proof-of-concept application: Tuto-U

Fig. 5.3 represents the current application, which achieves perfect scores of 100 in all three metrics: Performance, Accessibility, and Best Practices. This impressive achievement reflects the application's adherence to the proposed framework, which integrates state-of-the-art technologies and methodologies. The current application utilizes Next.js for server-side rendering, ensuring fast load times. It is deployed on Vercel, a cloud platform that offers robust and scalable deployment solutions. The integration of comprehensive data analytics, user-centered design principles, and seamless data integration further enhances the application's performance and user experience.



91

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS

● First Contentful Paint 1.8 s	■ Largest Contentful Paint 2.5 s
● Total Blocking Time 40 ms	● Cumulative Layout Shift 0.019
▲ Speed Index 5.9 s	

Figure 5.4: Lighthouse results of older version of peer-tutoring application: OrientaYT

In contrast, the Fig. 5.4 represents an older version of the same application. This version scored 91 in Performance, 85 in Accessibility, and 68 in Best Practices. The older application architecture consisted of a separate React front-end application and an Express framework for a Node.js back-end application, both deployed on a physical Windows Server provided by the TICs Department of Yachay Tech University. This setup, while functional, did not leverage the full potential of modern web development frameworks and cloud-based infrastructure.

Key differences between the two versions include:

- **Technology Stack:** The current application employs Next.js, which offers superior performance benefits through server-side rendering. The older version used React for the front end and Express for the back end, without the advanced optimizations provided by Next.js.

- **Deployment:** The current application is deployed on Vercel, a cloud platform that provides seamless deployment, automatic scaling, and global content delivery. The older version relied on a Windows Server physical infrastructure, which lacked the flexibility and scalability of a cloud-based solution.

- **Performance Optimization:** The current application benefits from Next.js's optimizations, including faster initial load times and efficient handling of dynamic content. These enhancements are reflected in the perfect performance score. The older version, while still performant, could not match the efficiency of Next.js and Vercel.
- **Accessibility and Best Practices:** The current application's perfect scores in Accessibility and Best Practices indicate a thorough implementation of web standards and guidelines. This includes improved semantic HTML, ARIA roles, color contrast, and security practices. The older version, with lower scores, highlights areas where these standards were not fully met, such as less optimal color contrast and security measures.

In summary, the transition to the proposed framework, including the use of Next.js and Vercel, has significantly enhanced the web application's performance, accessibility, and adherence to best practices. These improvements demonstrate the effectiveness of modern web development technologies and methodologies in creating high-quality, user-centric web applications. The comparison underscores the importance of implementing advanced tools and cloud infrastructure to achieve optimal results in web development.

5.5 Alignment with Smart Campus Framework

This section analyzes the proof-of-concept application based on the Smart Campus Framework outlined in the State of the Art chapter. Each layer of the framework is examined with specific details from the proof-of-concept application, as illustrated in Figure 5.5.

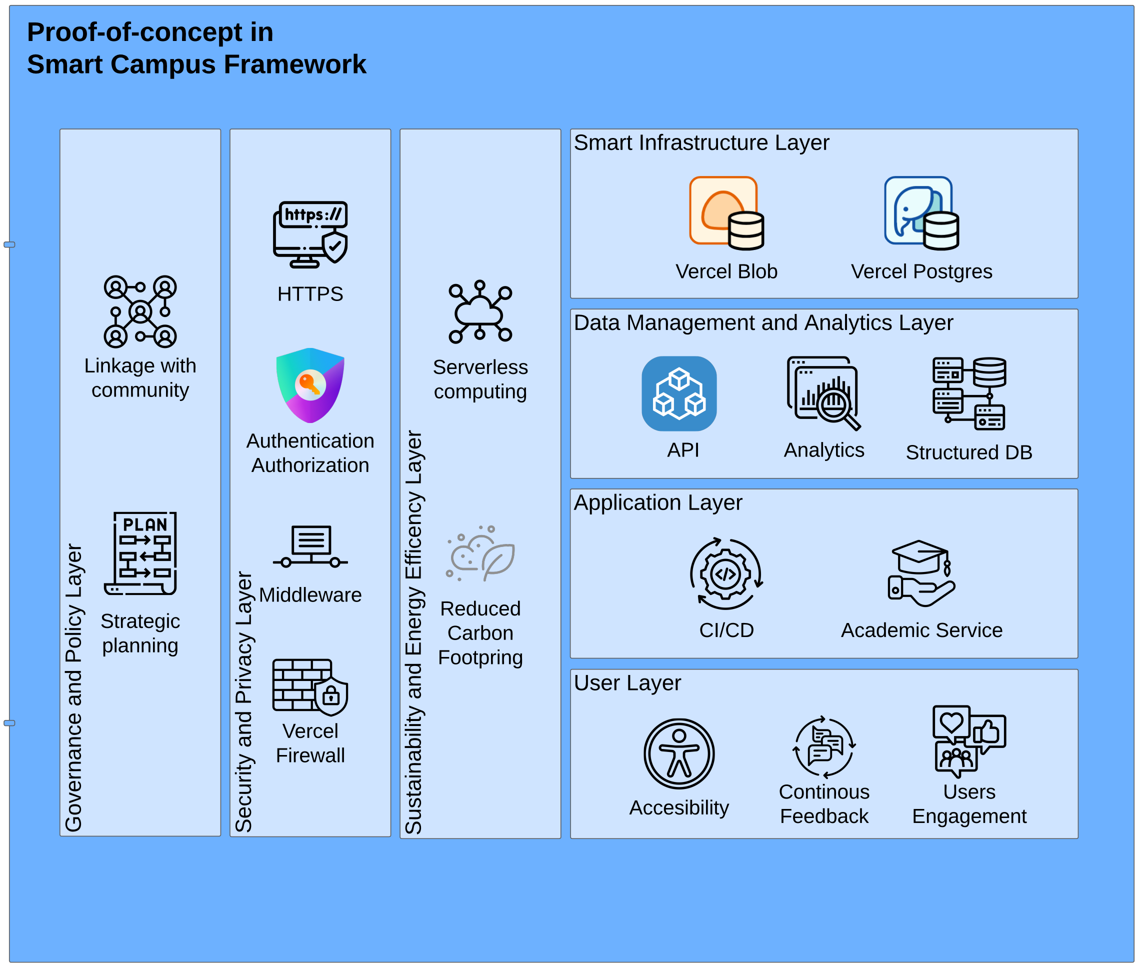


Figure 5.5: Proof-of-concept application in Smart Campus Framework.

5.5.1 Smart Infrastructure Layer

The foundation of the proof-of-concept application is built upon robust cloud services, prioritizing Vercel Postgres for database management and Vercel Blob for storage solutions. The use of these cloud services ensures scalability and reliability, essential components of the smart infrastructure layer. Vercel's infrastructure supports seamless deployment and high availability, aligning with the framework's requirement for a robust and flexible network infrastructure.

5.5.2 Data Management and Analytics Layer

Data management in the proof-of-concept application is streamlined using tRPC, which defines a clear and efficient API. This facilitates future integration with other systems and enables advanced data analysis. The application data is well-structured in a relational database, ensuring data integrity and accessibility. Additionally, Vercel's analytics tools provide valuable insights into user behavior and system performance, supporting data-driven decision-making and continuous improvement.

5.5.3 Application Layer

As an academic application, the proof-of-concept fits well within the Application Layer of the smart campus framework. The use of Vercel enhances the development and deployment process, allowing for rapid incorporation of new features and functionalities through its continuous integration and continuous deployment (CI/CD) pipelines. This ensures the application remains up-to-date and responsive to user needs, fostering a dynamic academic environment.

5.5.4 User Layer

The user experience is a core focus of the proof-of-concept application, designed with user-centered design principles. The application optimizes accessibility by adhering to WCAG guidelines, ensuring it is usable by individuals with diverse abilities. Participatory design practices were employed to gather continuous feedback from stakeholders, leading to an interface that meets user expectations and enhances satisfaction. Features such as session ratings and feedback mechanisms further engage users and promote a collaborative learning environment.

5.5.5 Security and Privacy Layer

Security and privacy are paramount in the proof-of-concept application, addressed through the implementation of NextAuth for authentication and authorization. The use of Next.js Middleware ensures role-based access control and secure redirects. Additionally, Vercel's Firewall provides automatic mitigation of DDoS attacks and allows for custom IP blocking rules, enhancing the overall security posture of the application and protecting user data. Furthermore, the web application utilizes HTTPS to ensure the confidentiality of user information during data transmission.

5.5.6 Sustainability and Energy Efficiency Layer

The proof-of-concept application makes use of cloud services and serverless computing, specifically Vercel infrastructure ensures that resources are used optimally which support efficient use of computational resources [112, 113]. This aligns with the sustainability goals of the smart campus framework, promoting energy efficiency and reducing the carbon footprint.

5.5.7 Governance and Policy Layer

The deployment and integration of the proof-of-concept application were guided by strategic planning initiatives, specifically a pilot plan developed as a project of linkage with the community. This aligns with Yachay Tech University's goals of community engagement and social responsibility. The application supports the strategic objectives of the campus, demonstrating the potential for broader implementation and integration within the university's governance framework. By aligning the proof-of-concept application with the Smart Campus Framework, we ensure that it not only meets current needs but is also scalable, sustainable, and secure, supporting the continuous evolution of smart campus environments.

In conclusion, the alignment of the proof-of-concept application with the Smart Campus Framework demonstrates the effectiveness of the proposed web application development framework. Each layer of the Smart Campus Framework was meticulously addressed, ensuring robust connectivity, efficient data management, user-centric design, and stringent security measures. The implementation of these layers within the proof-of-concept application showcases how a well-structured framework can optimize the functionality and user experience in a smart campus environment. The successful implementation of the proof-of-concept application validates the practicality and scalability of the framework, paving the way for future advancements in smart campus technologies. This alignment highlights the potential for significant improvements in campus operations and user satisfaction through the adoption of smart technologies and well-structured development frameworks. As educational institutions continue to evolve, the integration of such frameworks will be fundamental in creating intelligent, responsive, and sustainable campus environments that satisfy the diverse needs of their communities. The results and discussions presented in this chapter underscore the importance of a holistic approach to web application development for smart campuses, emphasizing the need for continuous innovation and adaptation to emerging technologies and user needs.

Chapter 6

Conclusions

Throughout this thesis, we have conducted a thorough investigation into the development of web applications specifically designed for smart campus environments. Our research has entailed an in-depth analysis of current technologies, the creation and deployment of innovative frameworks, and the assessment of their impact on optimizing campus functionalities and enriching educational experiences. We highlight the significant findings and culminate the chapter by proposing potential avenues for future research within this domain.

6.1 Conclusions

1. We conducted an exhaustive literature review, tracing the evolution from conceptual frameworks to the implementation of smart campus applications. This review is a fundamental first step for understanding the current landscape, identifying prominent trends. It provided a solid background for the subsequent developmental work, highlighting the need for innovative solutions that address identified challenges.

2. The research has introduced a scalable architecture tailored to meet the dynamic requirements of smart campuses. This architecture prioritizes adaptability and user experience, ensuring that web applications can effectively evolve in response to the expanding and evolving needs of the campus community. By emphasizing these principles, the development represents a significant advancement in the field, offering a flexible and robust framework capable of supporting diverse applications and services on smart campuses.

3. We successfully developed a web application for student-to-student tutoring as a proof-of-concept for the proposed framework. The application demonstrates the potential to significantly enhance the smart campus model by improving accessibility and the overall user experience in managing tutoring services. Our findings suggest

that integrating such technology can streamline tutoring processes, making educational support more efficient and readily available.

4. The design of the data integration component within the proposed framework has significantly enhanced its analytical capabilities, enabling data-driven decision-making processes. This enhancement is crucial for optimizing campus operations and educational strategies. By facilitating the seamless integration and analysis of diverse data sources, the framework empowers stakeholders to make informed decisions. The improved data analytical capabilities ensure that the campus community can effectively leverage data to drive strategic initiatives and improve overall efficiency.

5. We successfully designed a user interface for the web application that is both attractive and accessible. By emphasizing the effective integration of User-Centered Design (UCD) principles during the proof-of-concept development process, we ensured that the application is intuitive and focused on the needs of the campus community. Our design process centered on understanding the needs and preferences of students, faculty, and administrative staff, resulting in an interface that fosters positive interactions and satisfaction.

6.2 Future Work

The research presented in this thesis focuses on the design and development of web applications tailored for smart campus environments. Nevertheless, there is ample opportunity for future exploration and enhancement in this direction. Some potential avenues for future work include:

1. Building upon the foundation laid by the integration of cloud technology in the development of smart campus environments, future research could explore several promising avenues for further advancement. One potential area of focus is the refinement and optimization of cloud-based services to better cater to the diverse needs of teachers, students, and administrators. This could involve the development of customized applications and tools tailored to specific educational workflows and requirements, thereby enhancing the overall user experience and productivity within the smart campus ecosystem.

2. Additionally, there is potential for further exploration into the integration of emerging technologies, such as artificial intelligence and Internet of Things (IoT), with cloud-based smart campus solutions. By leveraging AI algorithms and IoT devices, smart campuses can achieve greater levels of automation, efficiency, and personalization in various aspects of campus operations and services. Future research could delve into the development of AI-driven analytics platforms for optimizing resource allocation,

predictive maintenance systems for campus infrastructure, and personalized learning environments for students.

3. Finally, as smart campuses continue to evolve and adapt to changing educational paradigms, ongoing evaluation and refinement of cloud-based solutions will be essential. Future research could involve longitudinal studies and user feedback mechanisms to assess the effectiveness and usability of cloud-based services over time. By incorporating insights from stakeholders, including students, faculty, and administrators, researchers can iteratively improve and optimize smart campus environments to meet the evolving needs of the educational community.

By pursuing these hints for future work, researchers can contribute to the ongoing evolution and improvement of smart campus environments, ultimately enhancing the educational experience and operational efficiency within higher education institutions.

Bibliography

[1] V. Ahmed, K. Abu Alnaaj, and S. Saboor, “An investigation into stakeholders’ perception of smart campus criteria: The american university of sharjah as a case study,” *Sustainability*, vol. 12, no. 12, p. 5187, 2020.

[2] N. Chagnon-Lessard, L. Gosselin, S. Barnabé, T. Bello-Ochende, S. Fendt, S. Goers, L. C. P. D. Silva, B. Schweiger, R. Simmons, A. Vandersickel, and P. Zhang, “Smart campuses: Extensive review of the last decade of research and current challenges,” *IEEE Access*, vol. 9, pp. 124 200–124 234, 2021.

[3] W. Villegas-Ch, J. Molina-Enriquez, C. Chicaiza-Tamayo, I. Ortiz-Garcés, and S. Luján-Mora, “Application of a big data framework for data monitoring on a smart campus,” *Sustainability*, vol. 11, no. 20, p. 5552, 2019.

[4] G. Ikrissi and T. Mazri, “A study of smart campus environment and its security attacks,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 44, pp. 255–261, 2020.

[5] Y. Eshet, “Digital literacy: A conceptual framework for survival skills in the digital era,” *Journal of Educational Multimedia and Hypermedia*, vol. 13, no. 1, pp. 93–106, January 2004. [Online]. Available: https://www.learntechlib.org/p/4793

[6] P. S. Leeflang, P. C. Verhoef, P. Dahlström, and T. Freundt, “Challenges and solutions for marketing in a digital era,” *European Management Journal*, vol. 32, no. 1, pp. 1–12, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263237313001576

[7] A. Kirkwood and L. Price, “Examining some assumptions and limitations of research on the effects of emerging technologies for teaching and learning in higher education,” *British Journal of Educational Technology*, vol. 44, no. 4, pp. 536–543, 2013. [Online]. Available: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12049

[8] R. M. Davison, C. Wagner, and L. C. Ma, “From government to e-government: a transition model,” *Information technology & people*, vol. 18, no. 3, pp. 280–299, 2005.

[9] L. Luo, “Data acquisition and analysis of smart campus based on wireless sensor,” *Wireless Personal Communications*, vol. 102, pp. 2897–2911, 2018.

[10] E. Gilman, S. Tamminen, R. Yasmin, E. Ristimella, E. Peltonen, M. Harju, L. Lovén, J. Riekki, and S. Pirttikangas, “Internet of things for smart spaces: A university campus case study,” *Sensors*, vol. 20, no. 13, 2020.

[11] A. H. Celdrán, F. J. G. Clemente, J. Saenz, L. De La Torre, C. Salzmann, and D. Gillet, “Self-organized laboratories for smart campus,” *IEEE Transactions on Learning Technologies*, vol. 13, no. 2, pp. 404–416, 2020.

[12] X. Xu, Y. Wang, and S. Yu, “Teaching performance evaluation in smart campus,” *IEEE Access*, vol. 6, pp. 77 754–77 766, 2018.

[13] A. de Paola, A. Giammanco, G. lo Re, and G. Anastasi, “Detection of points of interest in a smart campus,” in *2019 IEEE 5th International forum on Research and Technology for Society and Industry (RTSI)*, 2019, pp. 155–160.

[14] N. Min-Allah and S. Alrashed, “Smart campus—a sketch,” *Sustainable Cities and Society*, vol. 59, p. 102231, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670720302183

[15] T. Omotayo, A. Moghayedi, B. Awuzie, and S. Ajayi, “Infrastructure elements for smart campuses: A bibliometric analysis,” *Sustainability*, vol. 13, no. 14, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/14/7960

[16] A. K. Bart Valks, Monique H. Arkesteijn and A. C. den Heijer, “Towards a smart campus: supporting campus decisions with internet of things applications,” *Building Research & Information*, vol. 49, no. 1, pp. 1–20, 2021.

[17] R. Jurva, M. Matinmikko-Blue, V. Niemelä, and S. Nenonen, “Architecture and operational model for smart campus digital infrastructure,” *Wireless Personal Communications*, vol. 113, pp. 1437–1454, 2020.

[18] W. Li, “Design of smart campus management system based on internet of things technology,” *Journal of Intelligent & Fuzzy Systems*, vol. 40, no. 2, pp. 3159–3168, 2021.

[19] W. Muhamad, N. B. Kurniawan, Suhardi, and S. Yazid, “Smart campus features, technologies, and applications: A systematic literature review,” in *2017 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2017, pp. 384–391.

[20] K. AbuAlnaaj, V. Ahmed, and S. Saboor, “A strategic framework for smart campus,” in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, vol. 22, 2020, pp. 790–798.

[21] R. Vyas, “Comparative analysis on front-end frameworks for web applications,” *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 7, pp. 298–307, 2022.

[22] F. Almeida and J. Monteiro, “The role of responsive design in web development.” *Webology*, vol. 14, no. 2, 2017.

[23] P. C. van Oorschot and P. C. van Oorschot, “Web and browser security,” *Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin*, pp. 245–279, 2021.

[24] J. A. Hoffer, V. Ramesh, and H. Topi, *Modern database management.* Pearson, 2016.

[25] D. A. Norman and S. W. Draper, *User Centered System Design; New Perspectives on Human-Computer Interaction.* USA: L. Erlbaum Associates Inc., 1986.

[26] M.-K. D. Abras C. and P. J., “User-centered design,” *Encyclopedia of Human-Computer Interaction.*, vol. 34, no. 4, pp. 119–135, 2004.

[27] D. Norman, *The desing of every things.* Basic books, New York, 1988.

[28] J. Dumas and J. Redish, “A practical guide to usability testing,” 1993. [Online]. Available: https://api.semanticscholar.org/CorpusID:108979146

[29] E. Björgvinsson, P. Ehn, and P.-A. Hillgren, “Participatory design and” democratizing innovation”,” in *Proceedings of the 11th Biennial participatory design conference*, 2010, pp. 41–50.

[30] F. Kensing and J. Blomberg, “Participatory design: Issues and concerns,” *Computer supported cooperative work (CSCW)*, vol. 7, no. 3-4, pp. 167–185, 1998.

[31] T. Keinonen, “User-centered design and fundamental need,” in *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, 2008, pp. 211–219.

[32] C. McKnight, A. Dillon, and J. Richardson, *User centered design of hypertext and hypermedia for education.* New York: Macmillan, 1996. [Online]. Available: http://hdl.handle.net/10150/106501

[33] V. J. Traver, "Can user-centered interface design be applied to education?" *ACM SIGCSE Bulletin*, vol. 39, no. 2, pp. 57–61, 2007.

[34] P. Vassiliadis, "A survey of extract–transform–load technology," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 5, no. 3, pp. 1–27, 2009.

[35] A. A. Yulianto, "Extract transform load (etl) process in distributed database academic data warehouse," *APTIKOM Journal on Computer Science and Information Technologies*, vol. 4, no. 2, pp. 61–68, 2019.

[36] D. A. Chappell, *Enterprise service bus: Theory in practice.* " O'Reilly Media, Inc.", 2004.

[37] B. Suzic, "User-centered security management of api-based data integration workflows," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium.* IEEE, 2016, pp. 1233–1238.

[38] R. Van Der Lans, *Data Virtualization for business intelligence systems: revolutionizing data integration for data warehouses.* Elsevier, 2012.

[39] M. Frehner and M. Brändli, "Virtual database: Spatial analysis in a web-based data management system for distributed ecological data," *Environmental Modelling & Software*, vol. 21, no. 11, pp. 1544–1554, 2006.

[40] E. B. Mandinach, M. Honey, and D. Light, "A theoretical framework for data-driven decision making," in *annual meeting of the American Educational Research Association, San Francisco, CA*, 2006.

[41] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big data*, vol. 1, no. 1, pp. 51–59, 2013.

[42] E. Brynjolfsson and K. McElheran, "Data in action: data-driven decision making and predictive analytics in us manufacturing," *Rotman School of Management Working Paper*, no. 3422397, 2019.

[43] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," *Software Development*, vol. 9, no. 2001, pp. 28–35, 2001.

[44] J. Sutherland and K. Schwaber, "The scrum guide," *The definitive guide to Scrum: The rules of the game*, 2013.

[45] K. Beck, *Extreme Programming Explained: Embrace Change.* Addison-Wesley Professional, 2001.

[46] S. W. Ambler and M. Lines, *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise.* IBM press, 2012.

[47] K. Schwaber, "Scrum development process," in *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas.* Springer, 1997, pp. 117–134.

[48] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," in *2013 39th Euromicro conference on software engineering and advanced applications.* IEEE, 2013, pp. 9–16.

[49] C. Prandi, L. Monti, C. Ceccarini, and P. Salomoni, "Smart campus: Fostering the community awareness through an intelligent environment," *Mobile Networks and Applications*, vol. 25, pp. 945–952, 2020.

[50] N. Verstaevel, J. Boes, and M.-P. Gleizes, "From smart campus to smart cities issues of the smart revolution," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2017, pp. 1–6.

[51] N. Khamis and K. K. K. Li, "User experience evaluation for a bus tracking apps in smart campus initiative," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2254–2262, 2021.

[52] A.-M. Yang, S.-S. Li, C.-H. Ren, H.-X. Liu, Y. Han, and L. Liu, "Situational awareness system in the smart campus," *Ieee Access*, vol. 6, pp. 63 976–63 986, 2018.

[53] Y. Zhang, C. Yip, E. Lu, and Z. Y. Dong, "A systematic review on technologies and applications in smart campus: A human-centered case study," *IEEE Access*, vol. 10, pp. 16 134–16 149, 2022.

[54] N. N. M. Kasim and F. Khalid, "Choosing the right learning management system (lms) for the higher education institution context: A systematic review." *International Journal of Emerging Technologies in Learning*, vol. 11, no. 6, 2016.

[55] M. F. Paulsen, "Experiences with learning management systems in 113 european institutions," *Journal of Educational Technology & Society*, vol. 6, no. 4, pp. 134–148, 2003.

[56] C. C. Aydin and G. Tirkes, "Open source learning management systems in e-learning and moodle," in *IEEE EDUCON 2010 Conference.* IEEE, 2010, pp. 593–600.

[57] H. Nguyen, H. Zhao, S. Jamonnak, J. Kilgallin, and E. Cheng, "Rooway: A web-based application for ua campus directions," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2015, pp. 362–367.

[58] R. E. Roth, J. Van Den Hoek, A. Woodruff, A. Erkenswick, E. McGlynn, and J. Przybylowski, "The 21st century campus map: Mapping the university of wisconsin-madison," *Journal of Maps*, vol. 5, no. 1, pp. 1–8, 2009.

[59] H. C. Chan and L. Chan, "Smart library and smart campus," *Journal of Service Science and Management*, vol. 11, no. 6, pp. 543–564, 2018.

[60] J. Suhartono, S. Karya, and S. Candra, "The utilize of nfc technology for campus library services management," in *2017 International Conference on Information Management and Technology (ICIMTech)*. IEEE, 2017, pp. 60–64.

[61] H. Peng, "Research on the integration interface techniques for library management system and campus smart card system," in *2009 International Workshop on Intelligent Systems and Applications*. IEEE, 2009, pp. 1–4.

[62] B. R. Oderuth, K. Ramkissoon, and R. K. Sungkur, "Smart campus library system," in *2019 Conference on Next Generation Computing Applications (NextComp)*. IEEE, 2019, pp. 1–6.

[63] G. T. Evans and A. Beilby, "A library management information system in a multi-campus environment," *Clinic on Library Applications of Data Processing (19th: 1982)*, 1982.

[64] E. Bayangan-Cosidon, "Student information system for kalinga state university-rizal campus," *Journal of Management and Commerce Innovations*, vol. 4, no. 1, pp. 330–335, 2016.

[65] J. A. Ampofo, "Challenges of student management information system (mis) in ghana: A case study of university for development studies, wa campus," *International Journal of Management & Entrepreneurship Research*, vol. 2, no. 5, pp. 332–343, 2020.

[66] W. Han, S. Ada, R. Sharman, and H. R. Rao, "Campus emergency notification systems," *Mis Quarterly*, vol. 39, no. 4, pp. 909–930, 2015.

[67] M. Mohammed, K. Elleithy, and W. Elmannai, "Kmsafe app: Campus safety mobile app," in *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART)*. IEEE, 2021, pp. 1–4.

[68] J. Elliott, H. Jayachandran, P. Kumar, and K. Metzer, “Campus shuttle: Design of a college campus parking and transportation system,” in *2013 IEEE Systems and Information Engineering Design Symposium.* IEEE, 2013, pp. 104–109.

[69] S. A. Saad, M. H. I. Ishak, M. H. M. Fauzi, M. A. Baharudin, N. H. Idris *et al.*, “Real-time on-campus public transportation monitoring system,” in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA).* IEEE, 2018, pp. 215–220.

[70] W.-J. Shyr, L.-W. Zeng, C.-K. Lin, C.-M. Lin, and W.-Y. Hsieh, “Application of an energy management system via the internet of things on a university campus,” *EURASIA Journal of Mathematics, Science and Technology Education*, vol. 14, no. 5, pp. 1759–1766, 2018.

[71] D. Kolokotsa, K. Gobakis, S. Papantoniou, C. Georgatou, N. Kampelis, K. Kalaitzakis, K. Vasilakopoulou, and M. Santamouris, “Development of a web based energy management system for university campuses: The camp-it platform,” *Energy and Buildings*, vol. 123, pp. 119–135, 2016.

[72] H. A. Widjaja, B. Sablan, K. L. M. Ulo, K. Phusavat, A. N. Hidayanto *et al.*, “The development of integrated career portal in university using agile methodology,” in *2017 International Conference on Information Management and Technology (ICIMTech).* IEEE, 2017, pp. 310–315.

[73] U. Bhagat, N. Gujar, and S. Patel, “Implementation of iot in development of intelligent campus lighting system using mesh network,” in *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT).* IEEE, 2018, pp. 251–256.

[74] G. Sivasankar, S. Balaji, and N. Vignesh, “Internet of things based smart students’ body temperature monitoring system for a safe campus,” in *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS).* IEEE, 2022, pp. 416–420.

[75] N. P. Sastra and D. M. Wiharta, “Environmental monitoring as an iot application in building smart campus of universitas udayana,” in *2016 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS).* IEEE, 2016, pp. 85–88.

[76] M. Musa, M. N. Ismail, and M. F. M. Fudzee, “A survey on smart campus implementation in malaysia,” *JOIV : International Journal on Informatics Visualization*, vol. 5, pp. 51–56, MAR 2021. [Online]. Available: https://dx.doi.org/10.30630/joiv.5.1.434

[77] L. for Kwok, “A vision for the development of i-campus,” *Smart Learning Environments*, vol. 2, p. Not available, JAN 2015. [Online]. Available: https://dx.doi.org/10.1186/s40561-015-0009-8

[78] P. Carrion and M. Quaresma, “Internet da coisas (iot): Definições e aplicabilidade aos usuários finais,” *Human Factors in Design*, vol. 8, pp. 49–66, MAR 2019. [Online]. Available: https://dx.doi.org/10.5965/2316796308152019049

[79] C. LINDSAY, S. P. OSBORNE, and S. BOND, “The ‘new public governance’ and employability services in an era of crisis: Challenges for third sector organizations in scotland,” *Public Administration*, vol. 92, pp. 192–207, AUG 2013. [Online]. Available: https://dx.doi.org/10.1111/padm.12051

[80] S. J. Piotrowski and G. G. V. Ryzin, “Citizen attitudes toward transparency in local government,” *The American Review of Public Administration*, vol. 37, pp. 306–323, SEP 2007. [Online]. Available: https://dx.doi.org/10.1177/0275074006296777

[81] L. Trakman, “Modelling university governance,” *Higher Education Quarterly*, vol. 62, pp. 63–83, JAN 2008. [Online]. Available: https://dx.doi.org/10.1111/j.1468-2273.2008.00384.x

[82] P. H. Cheong and P. Nyaupane, “Smart campus communication, internet of things, and data governance: Understanding student tensions and imaginaries,” *Big Data amp; Society*, vol. 9, p. 205395172210926, JAN 2022. [Online]. Available: https://dx.doi.org/10.1177/20539517221092656

[83] X. Xu, D. Li, M. Sun, S. Yang, S. Yu, G. Manogaran, G. Mastorakis, and C. X. Mavromoustakis, “Research on key technologies of smart campus teaching platform based on 5g network,” *IEEE Access*, vol. 7, pp. 20 664–20 675, 2019.

[84] S. H. Gill, M. A. Razzaq, M. Ahmad, F. M. Almansour, I. U. Haq, N. Jhanjhi, M. Z. Alam, and M. Masud, “Security and privacy aspects of cloud computing: a smart campus case study,” *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 117–128, 2022.

[85] W. Villegas-Ch, X. Palacios-Pacheco, and S. Luján-Mora, “Application of a smart city model to a traditional university campus with a big data architecture: A sustainable smart campus,” *Sustainability*, vol. 11, no. 10, p. 2857, 2019.

[86] H. Im, R. S. Srinivasan, D. Maxwell, R. L. Steiner, and S. Karmakar, “The impact of climate change on a university campus’ energy use: Use of machine learning and building characteristics,” *Buildings*, vol. 12, no. 2, p. 108, 2022.

[87] M. Lillstrang, M. Harju, G. del Campo, G. Calderon, J. Röning, and S. Tamminen, "Implications of properties and quality of indoor sensor data for building machine learning applications: two case studies in smart campuses," *Building and Environment*, vol. 207, p. 108529, 2022.

[88] C. Jui-Fa, L. Wei-Chuan, J. Chih-Yu, and H. Ching-Chung, "A chinese interactive feedback system for a virtual campus," in *Technologies Shaping Instruction and Distance Education: New Studies and Utilizations.* IGI Global, 2010, pp. 290–316.

[89] L. Zhang and W. Song, "Research on intrusion detection algorithm based on smart campus network security," in *Proceedings of the 2020 International Conference on Aviation Safety and Information Technology*, 2020, pp. 446–449.

[90] I. Kim and A. J. Lee, "" i know what you did last semester": Understanding privacy expectations and preferences in the smart campus," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–15.

[91] T. Anagnostopoulos, P. Kostakos, A. Zaslavsky, I. Kantzavelou, N. Tsotsolas, I. Salmon, J. Morley, and R. Harle, "Challenges and solutions of surveillance systems in iot-enabled smart campus: a survey," *IEEE Access*, vol. 9, pp. 131 926–131 954, 2021.

[92] M. A. Bouazzouni, E. Conchon, F. Peyrard, and P.-F. Bonnefoi, "Trusted access control system for smart campus," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld).* IEEE, 2016, pp. 1006–1012.

[93] Z. Ali, M. A. Shah, A. Almogren, I. Ud Din, C. Maple, and H. A. Khattak, "Named data networking for efficient iot-based disaster management in a smart campus," *Sustainability*, vol. 12, no. 8, p. 3088, 2020.

[94] Y. Niu, H. Jiang, B. Tian, H. Xiang, Y. Liu, X. Xia, and Y. Zhao, "An efficient access control scheme for smart campus," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 9, no. 6, pp. e5–e5, 2022.

[95] A. M. Eltamaly, M. A. Alotaibi, A. I. Alolah, and M. A. Ahmed, "Iot-based hybrid renewable energy system for smart campus," *Sustainability*, vol. 13, no. 15, p. 8555, 2021.

[96] H. A. Muqeet, H. Javed, M. N. Akhter, M. Shahzad, H. M. Munir, M. U. Nadeem, S. S. H. Bukhari, and M. Huba, "Sustainable solutions for advanced energy management system of campus microgrids: Model opportunities and future challenges," *Sensors*, vol. 22, no. 6, p. 2345, 2022.

[97] C. Gackenheimer and C. Gackenheimer, “What is react?” *introduction to react*, pp. 1–20, 2015.

[98] A. Fedosejev, *React. js essentials.* Packt Publishing Ltd, 2015.

[99] A. Banks and E. Porcello, *Learning React: functional web development with React and Redux.* ” O’Reilly Media, Inc.”, 2017.

[100] S. Aggarwal *et al.*, “Modern web-development using reactjs,” *International Journal of Recent Research Aspects*, vol. 5, no. 1, pp. 133–137, 2018.

[101] A. Rastogi, N. Swamy, C. Fournet, G. Bierman, and P. Vekris, “Safe & efficient gradual typing for typescript,” in *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2015, pp. 167–180.

[102] L. Fischer and S. Hanenberg, “An empirical investigation of the effects of type systems and code completion on api usability using typescript and javascript in ms visual studio,” *ACM SIGPLAN Notices*, vol. 51, no. 2, pp. 154–167, 2015.

[103] G. Richards, F. Zappa Nardelli, and J. Vitek, “Concrete types for typescript,” in *29th European Conference on Object-Oriented Programming (ECOOP 2015).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[104] Vercel, Inc., “Next.js documentation,” https://nextjs.org/docs, 2023, accessed: 2023-12-12.

[105] tRPC Contributors, “trpc documentation,” https://trpc.io/docs, 2024, accessed: 2024-01-04.

[106] K. Sandoval, “Using trpc for typescript-enabled apis,” *Nordic APIs*, April 2023. [Online]. Available: https://nordicapis.com/using-trpc-for-typescript-enabled-apis/

[107] Prisma, “Prisma documentation,” https://www.prisma.io/docs/, 2023, accessed: 2023-12-21.

[108] S. Chacon and B. Straub, *Pro Git.* Apress, 2014. [Online]. Available: https://git-scm.com/book/en/v2

[109] GitHub, Inc., “Github documentation,” https://docs.github.com/, accessed: 2023-12-20.

[110] Vercel, Inc., “Announcing v0: Generative ui,” https://vercel.com/blog/announcing-v0-generative-ui, 2023, accessed: 2023-12-16.

[111] G. Developers, “Lighthouse: Tools for web developers,” 2024, accessed: 2024-01-24. [Online]. Available: https://developers.google.com/web/tools/lighthouse

[112] Vercel, “What is vercel’s green energy policy?” 2024, accessed: 2024-03-01. [Online]. Available: https://vercel.com/guides/what-is-vercel-green-energy-policy

[113] A. W. Services, “The cloud - amazon sustainability,” 2024, accessed: 2024-03-01. [Online]. Available: https://sustainability.aboutamazon.com

Appendix A

User Interface Captures

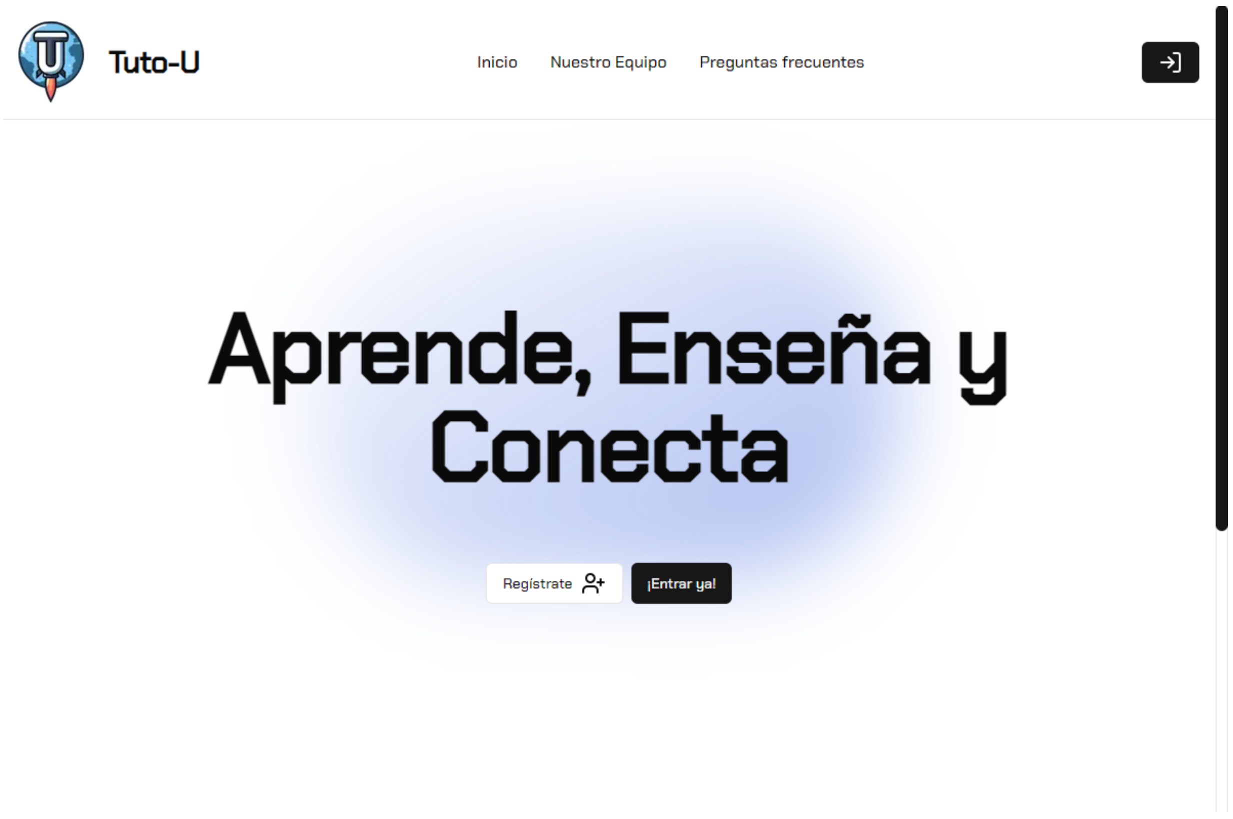


Figure A.1: Initial view of the application.

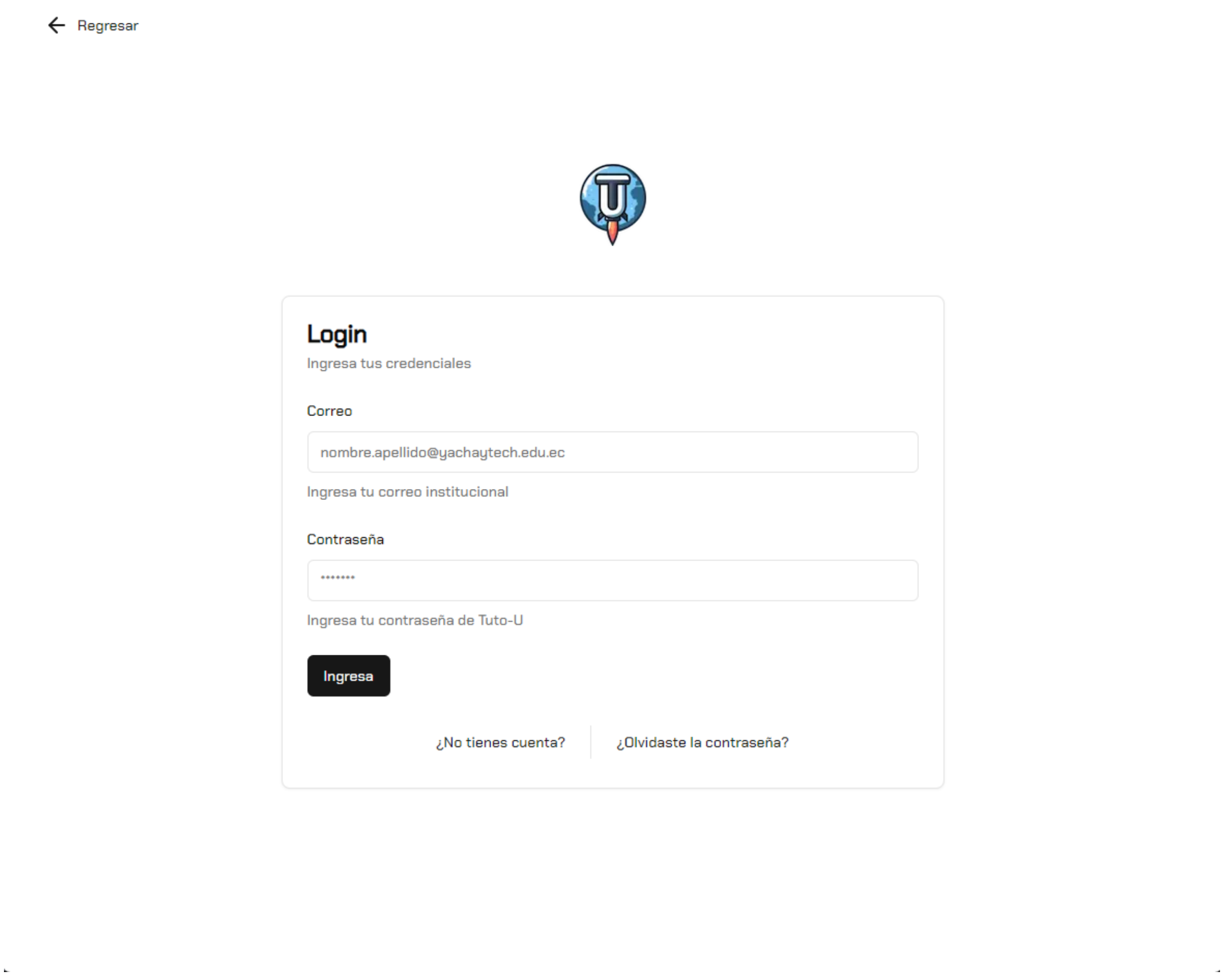

Regresar
Login
Ingresa tus credenciales
Correo
nombre.apellido@yachaytech.edu.ec
Ingresa tu correo institucional
Contraseña
•••••••
Ingresa tu contraseña de Tuto-U
Ingresa
¿No tienes cuenta?
¿Olvidaste la contraseña?

Figure A.2: Login view.

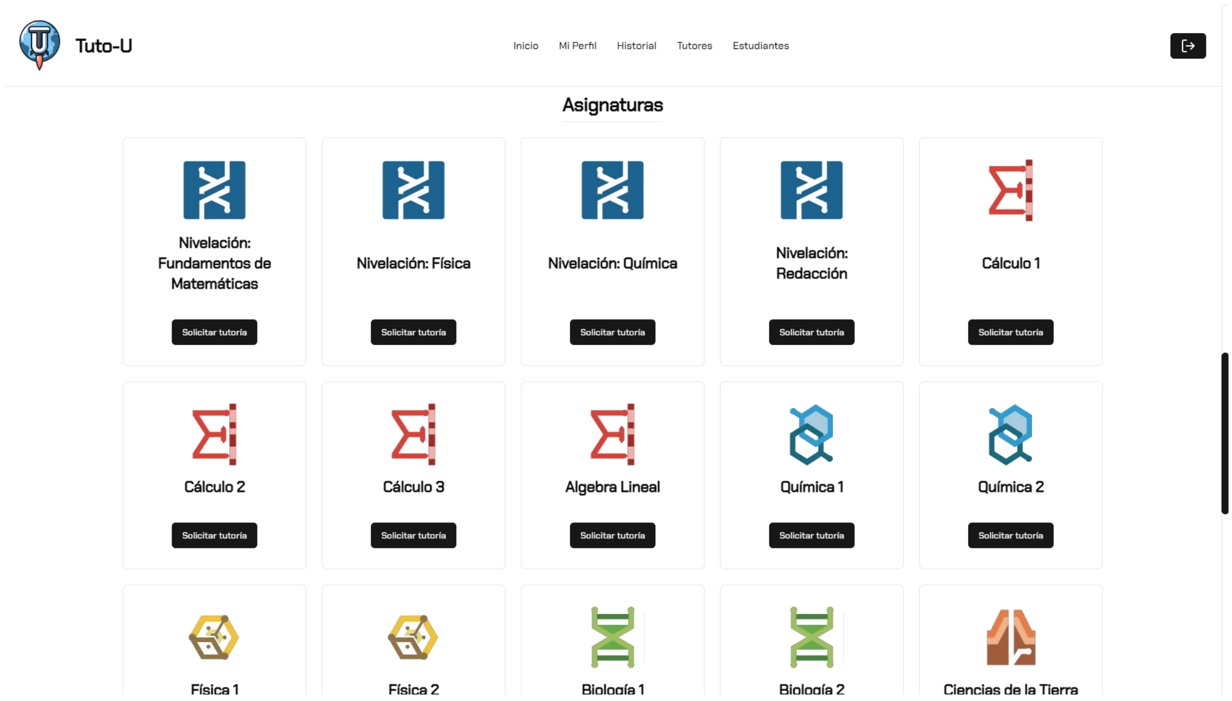
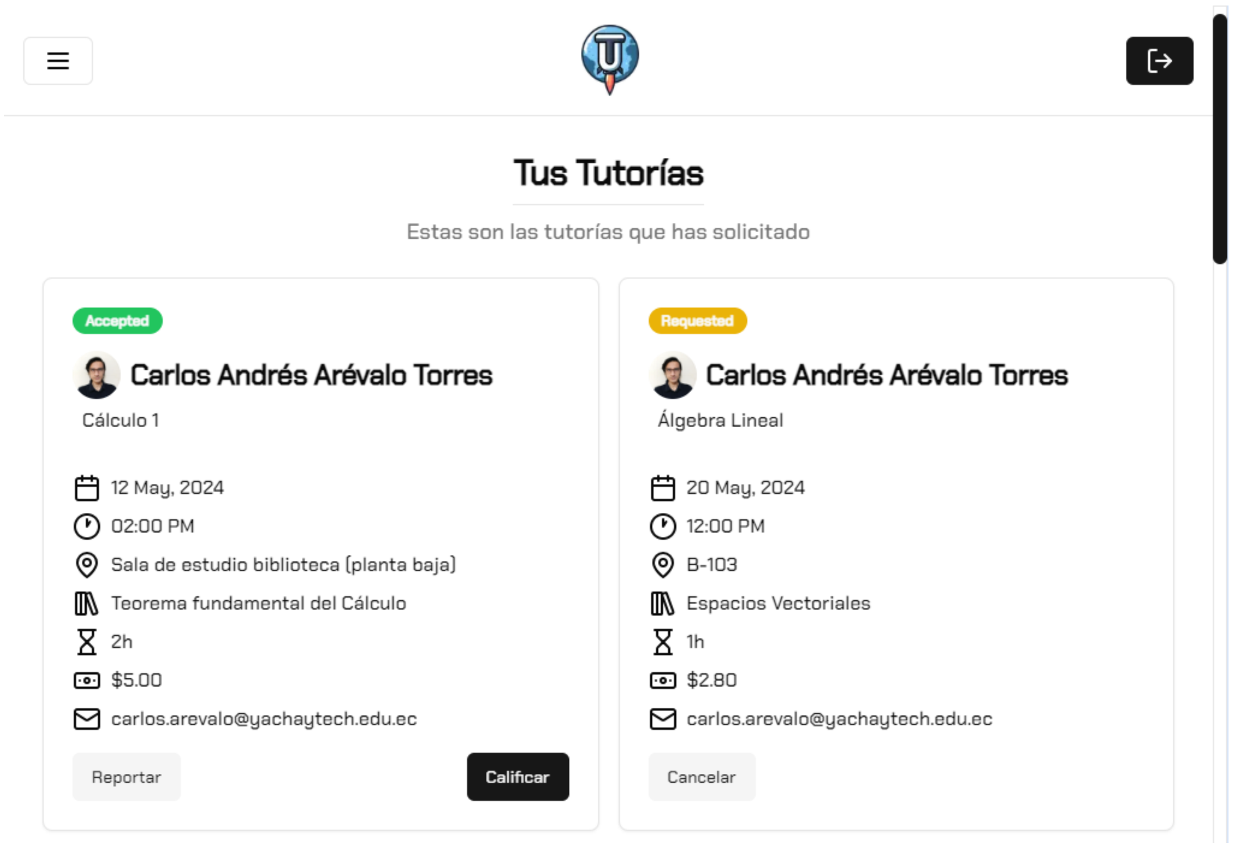


Figure A.3: Assignments view.



Tus Tutorías

Estas son las tutorías que has solicitado

Accepted

Carlos Andrés Arévalo Torres

Cálculo 1

12 May, 2024

02:00 PM

Sala de estudio biblioteca (planta baja)

Teorema fundamental del Cálculo

2h

$5.00

carlos.arevalo@yachaytech.edu.ec

Requested

Carlos Andrés Arévalo Torres

Álgebra Lineal

20 May, 2024

12:00 PM

B-103

Espacios Vectoriales

1h

$2.80

carlos.arevalo@yachaytech.edu.ec

Figure A.4: Session requests view.

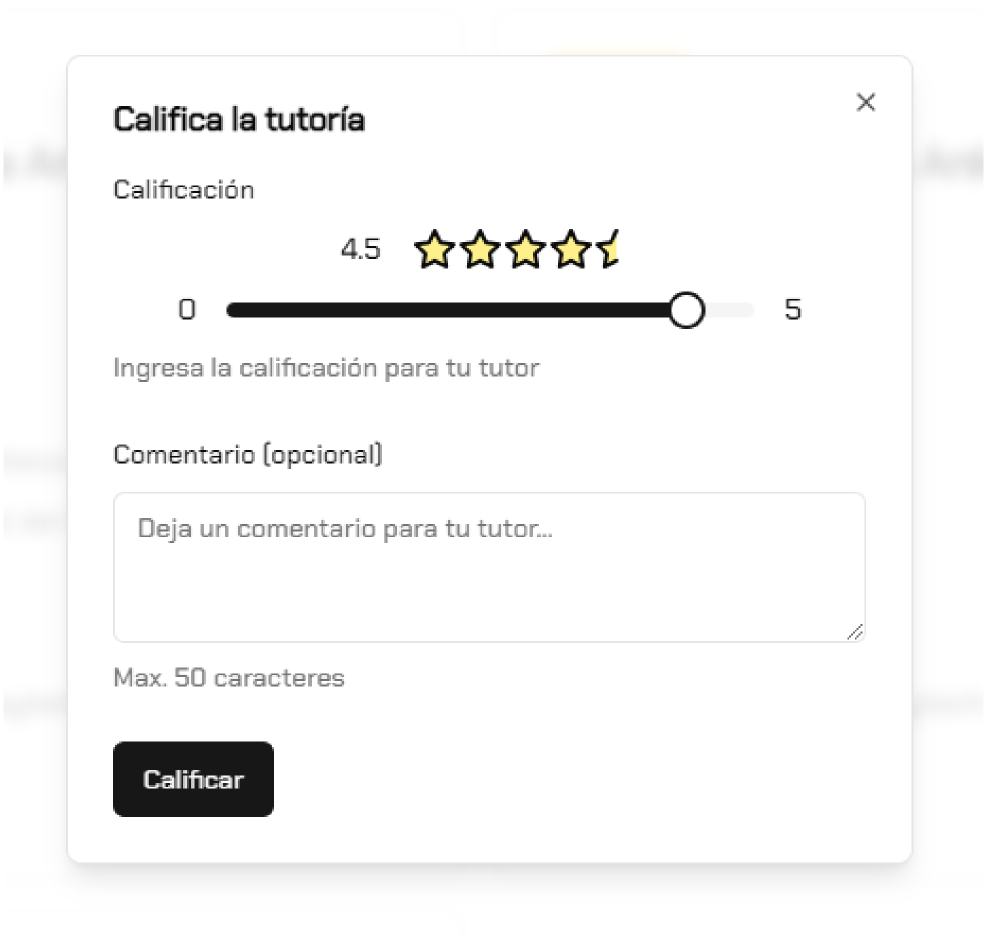


Figure A.5: Rating and feedback system.

Appendix B

Application database schema

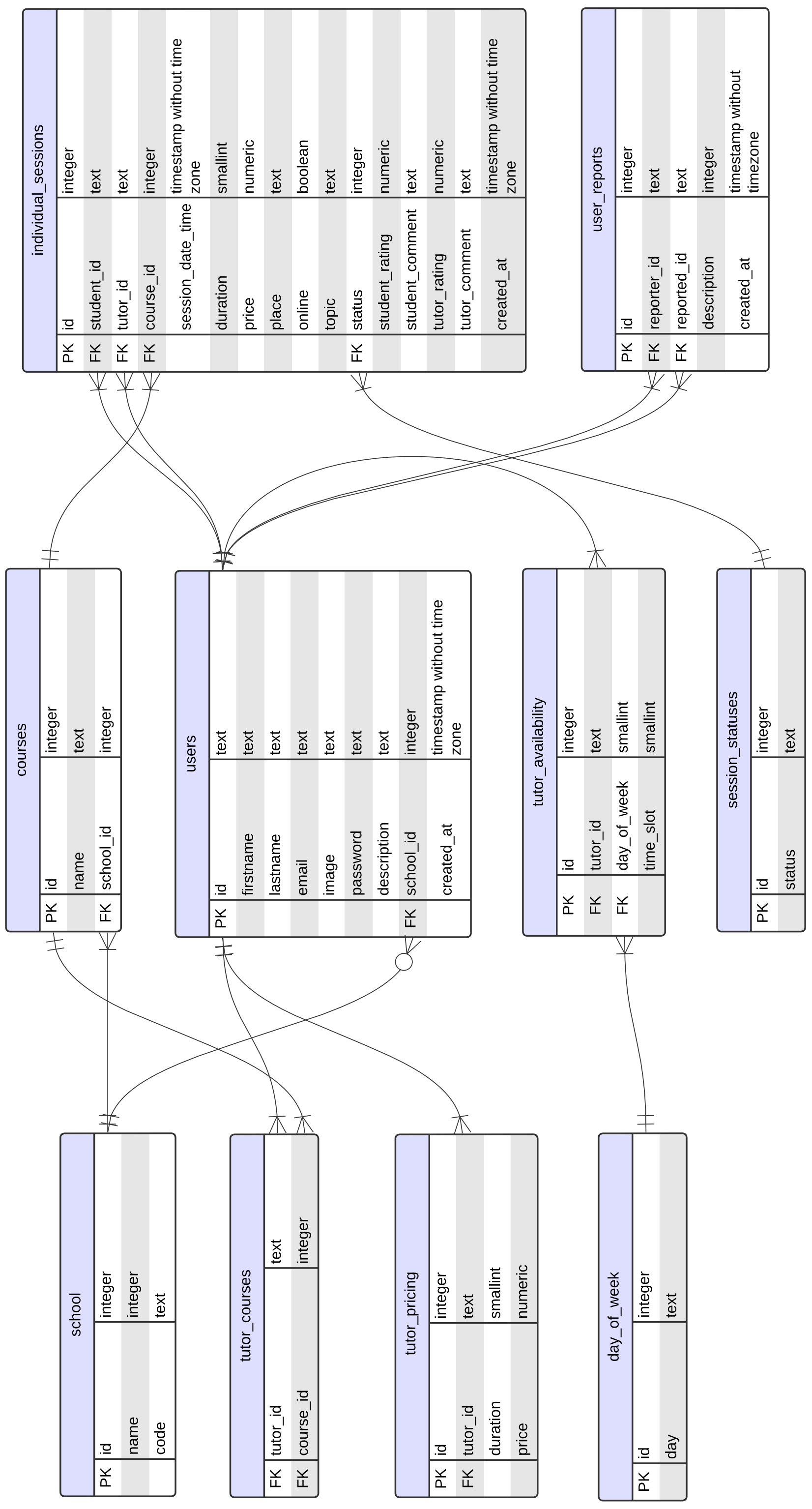


Figure B.1: Proof-of-concept application database schema.