Northumbria ISOC: A digital solution for the Muslim community at Northumbria University

1. Abstract

As digital transformation reshapes academic environments, the Northumbria ISOC app emerges as a tailored digital solution for Muslim students and staff at Northumbria University. This app aims to facilitate religious adherence and community engagement by integrating essential features such as synchronized prayer times with local mosques, digital Quran access, and an events calendar focused on ISOC committee activities. An innovative 'Ask Me' feature leverages the OpenAI API to address a broad range of queries, enhancing both the academic and social experience. Predictive simulations based on existing applications indicate that the app will significantly improve religious engagement and community integration. The design emphasizes user-friendliness and personalization, crucial for fostering frequent use and reliance. Preliminary evaluations suggest that the app will substantially enhance the practical and spiritual lives of its users, potentially setting new standards for supporting religious and cultural diversity in educational institutions.

Contents

2. Introduction

In the vibrant academic setting of Northumbria University, integrating student life with digital solutions is pivotal, especially for enhancing community and religious engagement among the Muslim population. The university provides physical prayer facilities and communicates Islamic events primarily through social media, which can lead to information being overlooked. This fragmented approach highlights the need for a more integrated and organized system.

The Northumbria ISOC app aims to consolidate essential features into a unified digital hub, including accurate prayer times, a digital Quran, and a well-organized calendar of ISOC events. By simplifying access to these resources, the app is designed to support religious observance and enhance community interaction among Muslim students.

The app's development was informed by extensive competitor analysis, scholarly research, and collaborative consultations with the Islamic society committee. Its anticipated impact goes beyond functionality; it is expected to revolutionize how Muslim students access and engage with their religious and cultural resources.

This report will narrate the journey of the Northumbria ISOC app from concept to anticipated deployment, detailing the research and planning that informed its features, the practical work involved in its development, and comprehensive evaluations of its functionality and user interface. It concludes with discussions on findings and recommendations for future enhancements or related projects, framing the app as a significant enhancement to the university experience for Northumbria's Muslim community.

3. Research and Planning

This section encapsulates the foundational work and meticulous strategies behind the development of the Northumbria ISOC app. It begins with an in-depth literature review that scrutinises existing research and industry practices, positioning the app within the current landscape of technological solutions that cater to religious practices and community engagement within educational settings. This comprehensive review sets the stage for the app's features, offering insights into their relevance and need.

Following the literature review, the methodology subsection delves into the systematic approach adopted for the app's development, detailing the use of the Agile project management framework. This approach, characterised by its iterative nature and emphasis on user feedback, is discussed alongside the planning, design, prototyping, and testing phases of the app. This narrative outlines how the app was shaped and refined in alignment with the real-world needs and expectations of Northumbria's Muslim students and staff, highlighting the thoughtful process behind creating an app that is both practical and engaging.

3.1 Literature Review

Introduction

Navigating university life as an international student involves balancing academic demands with the maintenance of cultural and religious identities. For Muslim students at Northumbria University, the challenge is profound, highlighting a distinct need for supportive digital tools. This literature review examines the potential of mobile technology to bridge this gap, underscoring the critical role it can play in supporting these students.

The focus is on conceptualising a mobile app tailored specifically for Northumbria's Muslim community. This app aims to simplify religious observance and enhance social integration by

providing organized access to prayer times, a digital Quran, and social event notifications. The review will first assess the broader use of mobile apps in academia to support student life and integration. It will then narrow to evaluate religious mobile apps, focusing on their design, impact, and the technologies that enhance their utility for Muslim students.

Critical discussions will include user-centred design principles ensuring the app's practicality and significance for its users. The review will also tackle developmental challenges and innovative solutions within mobile app technology, reflecting on how these align with user needs and technological advancements.

By mapping out the landscape of mobile technology in educational settings, this review aims not only to highlight existing research but also to identify gaps where further innovation could significantly impact the Muslim student community at Northumbria University. Through a structured exploration, it sets the groundwork for the forthcoming detailed analysis of mobile solutions designed to foster religious observance and community connectivity.

Mobile Apps in Academia

In the sphere of higher education, the integration of mobile applications has signified a profound shift in the academic and extracurricular engagement of students in recent years (GMI, 2019). A comprehensive study underscores the widespread use of technology during classroom activities, revealing that the digital integration of student life extends well beyond academic support to encompass social media and leisure applications. Predominantly used apps, such as Facebook, Google, and Instagram, dominate the digital landscape, as depicted in Figure 2. This pattern of usage, which varies depending on the context of the courses, highlights a flexible digital interaction, such as the notable preference for YouTube or Vimeo in design-related classes, shown in Figure 3 (Oliveira et al., 2021).

The discrepancy between student's self-perception and the actual usage of these apps, emphasised by (Oliveira et al., 2021), underscores the need for mobile applications that promote self-awareness and responsible usage, particularly during educational activities. This insight is particularly salient in the development of an app for the Muslim students at Northumbria University, as it must be attuned to these nuanced digital behaviours. The app's design must thus reflect an understanding of these patterns, encouraging engagement with educational and religious features during moments when students are most receptive, typically at the beginning and end of academic sessions, which can be strategically influenced by the application's notification system.

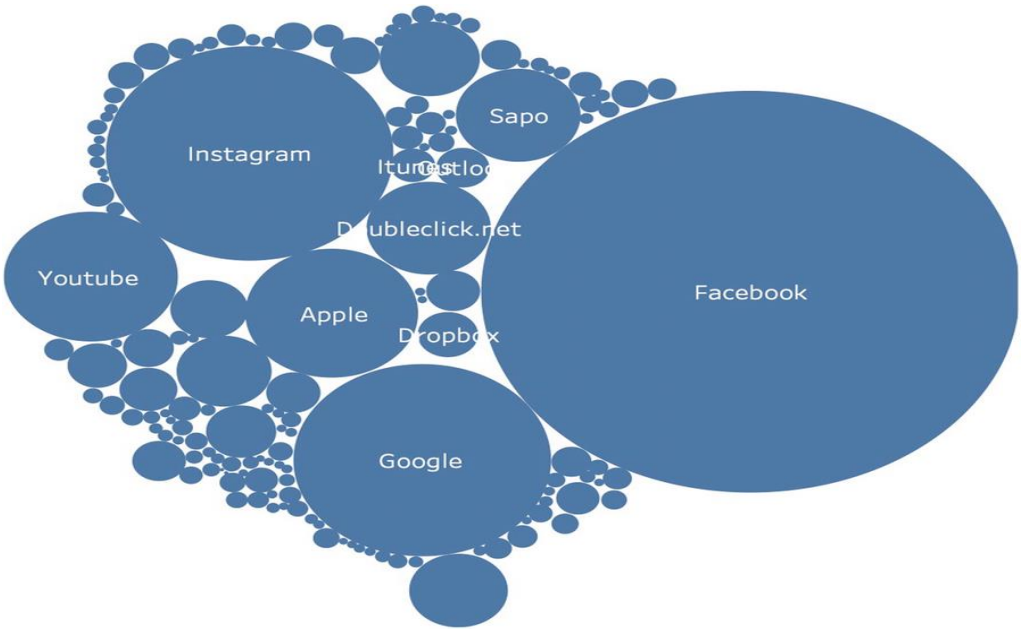


Figure 1. General use of applications in Scriptwriting class

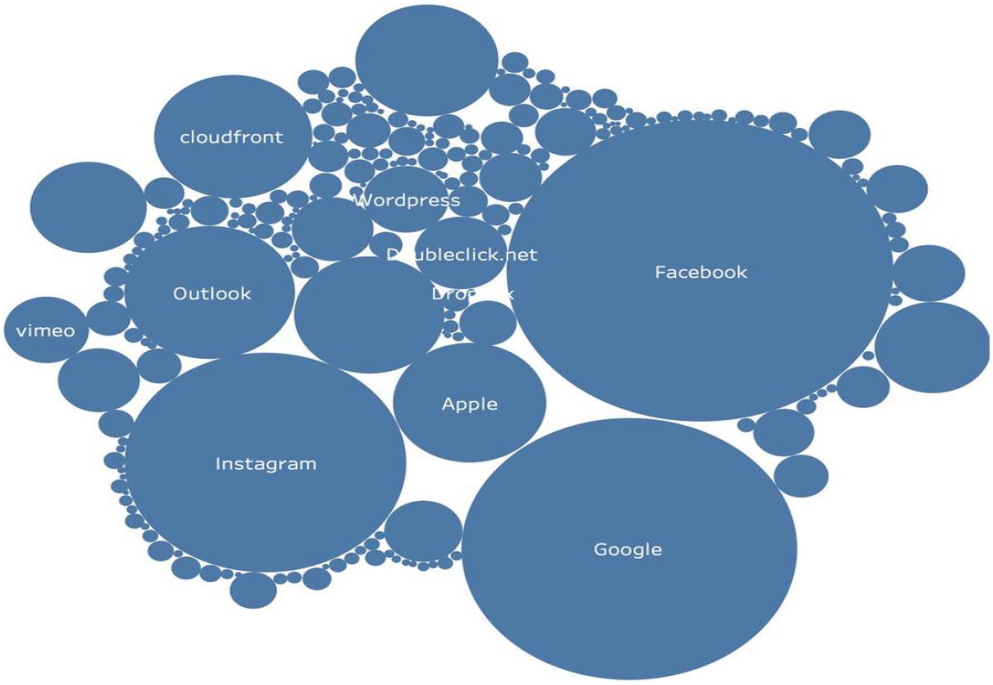


Figure 2. General use of applications in Aesthetics class

This approach to app development aligns the technological interactions of students with the functionalities that support their academic and religious commitments, facilitating a supportive digital environment for both educational and personal growth (Oliveira et al., 2021).

Further insights into mobile app adoption in academia emerge from a study on smart campus initiatives, which underscores the significance of the initial user experience. Features that are initially encountered by users, especially those readily accessible from the home screen, tend to dominate long-term use (Lumor et al., 2021). This finding highlights the importance of strategic feature placement and visibility in the app's interface, ensuring that essential resources for academic and religious observance are not only prominent but also inviting to explore. In addition to a compelling design, the study advocates for robust support mechanisms to encourage app adoption and continuity,

suggesting a blend of formal instructional support with informal, peer-led activities to foster a supportive digital ecosystem (Lumor et al., 2021).

The rise of mobile learning (m-learning) in higher education is transforming the student experience by facilitating individualised, flexible learning opportunities beyond the traditional classroom (Pechenkina, 2017). M-learning's ascent, characterised by the convenience of 'anytime, anywhere' access, aligns seamlessly with the trend of integrating mobile applications into academic contexts. This paradigm shift suggests the importance of creating apps that are not only multifunctional, accommodating a spectrum of educational activities, but also universally accessible to ensure inclusivity. Within this context, the project's use of React Native acknowledges the need for a deliberate design strategy that addresses these diverse educational requirements while ensuring that no student is disadvantaged due to accessibility barriers. To support the unique academic and religious commitments of Northumbria University's Muslim students, the app will be developed to align with (Pechenkina's, 2017) typology of HE apps, striving for a centralised platform that serves multiple purposes. By doing so, the app will serve as a comprehensive hub that simplifies students' management of their university life, underpinning a digitally supportive environment that caters to both their educational needs and personal development goals.

The collective evidence from these studies informs a forward-thinking approach to mobile app development within the academic sector. The envisioned app for Northumbria University's Muslim community is poised to address these nuanced facets of mobile app engagement, adoption, and accessibility. By thoughtfully integrating these considerations, the app aspires to enrich the academic journey and support the religious practices of its users, setting a benchmark for inclusivity and user-centred design in educational technology.

Religious Mobile Apps

The pervasive adoption of mobile technology has not only revolutionised everyday interactions but has profoundly influenced religious practices, particularly within the Muslim community. The extensive survey by Anum Hameed, Hafiza Anisa Ahmed, and Narmeen Zakaria Bawany critically evaluates about 300 Islamic mobile apps, shedding light on their pivotal roles and existing deficiencies (Hameed et al., 2019). These applications cater to fundamental religious needs, offering features like prayer times, Qibla directions, and Quranic access, fundamentally enhancing the observance of Islam in the digital age.

Hameed et al.'s study reveals a significant reliance on these apps among Muslims, as demonstrated by the high download rates of apps such as "Muslim Pro" (Hameed et al., 2019). These apps integrate essential religious practices with digital convenience, supporting users in maintaining their religious duties alongside their secular lives. This integration is crucial for students, who often struggle to balance their academic responsibilities with their religious commitments. The utility of these apps in facilitating religious practices aligns with findings from Campbell (Campbell, 2010) and Asadullah (Asadullah et al., 2014, pp. 45-49), who note the growing trend of digital platforms in enhancing religious education and practice.

Despite their benefits, the study identifies critical gaps in the current offerings of Islamic apps, particularly the issue of content authentication. Many apps disseminate incorrect or misleading information, a problem compounded by the lack of a dedicated "Religion" category in the Google Play Store (Hameed et al., 2019). This absence not only hinders the discoverability of authentic apps but also challenges users' ability to identify reliable sources of religious content. The concern over content authenticity is echoed in the broader discourse on digital religious practices, where scholars like Khan and Shambour (Khan & Shambour, 2017, pp. 37-47) emphasise the need for verified and authoritative content in religious apps.

Moreover, the user feedback and secondary literature reviewed within the study suggest a pressing demand for apps that go beyond basic functionalities to include advanced features like in-depth religious studies, interactive community platforms, and integration with educational systems (Hameed et al., 2019). This aligns with Mohamed et al.'s (Mohamed et al., 2016) research, which highlights the evolving expectations of users towards multifunctional religious apps that support diverse aspects of Muslim life, including education and social interaction.

In conclusion, the integration of religious practices with mobile technology, as detailed in the study by Hameed et al., presents both opportunities and challenges. While these apps significantly aid in the observance of religious duties, they also highlight the critical need for improved content verification, user-centred design, and educational integration. For institutions like Northumbria University, addressing these gaps could greatly enhance the religious and academic experiences of Muslim students, fostering a more inclusive and supportive educational environment. The insights from this study, bolstered by the referenced works within, provide a valuable framework for developing future apps that are both functional and trustworthy.

Technology in Religious Practices

The burgeoning field of "digital religion" highlights the profound impact that the internet and associated technologies have had on religious practices worldwide. The term, initially conceptualised by Foltz and Foltz (2003), refers to the transformation and mediation of religious activities through digital platforms. This transformation has facilitated the creation and sustenance of online religious communities, significantly altering traditional religious practices.

One of the pivotal changes brought about by digital technologies is the increased accessibility of religious content. Mobile apps, social media platforms, and live streaming services have become integral to religious observance, allowing adherents to access prayer times, religious texts, and virtual congregations at their convenience. This shift is not merely functional but deeply transformative, as it reshapes how individuals enact their beliefs and express their religious identities (Campbell and Rule, 2020).

However, the integration of technology into religious practice is not without its challenges and controversies. While digital tools can enhance participation, particularly among younger demographics more accustomed to digital interactions, they also raise questions about the authenticity and sanctity of online religious practices. Critics argue that the virtualisation of religious experiences might lead to a disembodied form of worship, where the physical and communal aspects of religious practice are diminished (Foltz and Foltz, 2003).

Moreover, there are ongoing debates within religious communities about the appropriateness of integrating technology into sacred traditions. These discussions often revolve around the potential commercialisation of religious practices, where spirituality may be seen as yet another consumer choice rather than a profound communal and personal commitment. The concern is that the sacredness of religious observances could be compromised by their transformation into digital commodities (Foltz and Foltz, 2003).

Despite these challenges, the benefits of digital religion are significant, particularly for individuals who may be physically unable to participate in traditional religious settings. For people with disabilities or those geographically isolated, digital platforms offer a means to engage with their faith that was previously unavailable. This inclusivity extends the reach of religious communities and allows for a more diverse expression of faith practices (Campbell, 2012).

As digital media continues to permeate all aspects of life, its influence on religious belief and community is expected to deepen, shaping contemporary and future conceptions of spirituality and communal religious life (Campbell and Rule, 2020). The ongoing evolution of digital religion

necessitates a critical examination of how technologies influence not just the logistics of religious practice, but also the very essence of spiritual experience and community building in the digital age.

User-Centred Design Principles

The concept of User-Centred Design (UCD) positions users at the forefront of the design process, integrating their needs and feedback to create more effective and accessible technologies (Norman, 2013). UCD emphasises understanding the user's background, preferences, and environment, which is fundamental in crafting solutions that are not only functional but also resonate with users on a personal level (Richardson et al., 1998).

Central to UCD is the focus on user goals, which extends beyond mere task completion to consider the broader context of user interactions. This principle fosters innovation by encouraging designers to conceptualise new ways to achieve user goals that may break from traditional approaches (Richardson et al., 1998). For instance, in software development, practical applications of UCD have shown to enhance user interaction significantly, as detailed by Lowdermilk (2013), who explores various metrics like emotional engagement and net promoter scores that gauge the effectiveness of user experience strategies.

Despite its advantages, UCD is not without challenges. A common critique is its potential overemphasis on usability, which could overshadow other design facets like aesthetics or innovation (Norman, 2013). Furthermore, the inclusivity of UCD processes can be complex, particularly when user groups are diverse and have conflicting needs. This complexity necessitates a careful balance in design decisions to accommodate varied user preferences without diluting the functionality or integrity of the product (Norman, 2013).

The adaptability of UCD principles across different industries illustrates their versatility. In healthcare, for example, UCD principles guide the design of medical devices to enhance usability and reduce errors, thereby improving patient safety. Similarly, in education, UCD can tailor educational technologies to fit diverse learning styles and needs, making learning more accessible and effective (Albert & Tullis, 2013).

As technology evolves, UCD faces new challenges and opportunities. Emerging technologies like AI, IoT, and mobile platforms offer novel ways to understand and cater to user needs. However, they also introduce issues related to privacy, security, and accessibility, which UCD must address to remain effective. These considerations are crucial as they ensure that technological advancements do not alienate users but rather enhance their interaction with technology in secure and meaningful ways (Albert & Tullis, 2013).

In conclusion, while UCD offers significant benefits in creating user-friendly designs, it requires continuous adaptation to meet the demands of advancing technology and changing user landscapes. By maintaining a flexible and holistic approach to UCD, designers can overcome its challenges and leverage its full potential to deliver solutions that are not only technically sound but also deeply attuned to the human aspects of user interaction.

Development Challenges and Innovations

In the evolving landscape of mobile application development, especially in domains centred around user engagement and religious technologies, developers encounter a variety of challenges and are continually innovating to overcome these obstacles. Technical challenges such as dealing with platform fragmentation, crafting intuitive and aesthetically pleasing user interfaces, and ensuring that religious texts and practices are appropriately integrated into applications are predominant (Lowdermilk, 2013; Foltz and Foltz, 2003).

The technical challenge of platform fragmentation, for instance, has been addressed through the adoption of cross-platform development frameworks like React Native, Flutter, or Xamarin. These frameworks allow for writing code once that can be executed on multiple platforms, thus reducing the complexity and cost associated with maintaining separate codebases for each platform (Lowdermilk, 2013). However, while these tools facilitate easier development across different platforms, they introduce new challenges such as maintaining performance and ensuring a uniform user experience across diverse device ecosystems.

In terms of user-centred design, the integration of skilled UI/UX designers in the development process has proven invaluable. These designers focus on understanding user needs and behaviours, conducting extensive user research, and developing interfaces that are not only functional but also engaging and easy to navigate. This collaboration is crucial in creating applications that are not just technologically sound but also align with user expectations and preferences, enhancing overall user satisfaction (Lowdermilk, 2013).

Regarding religious technologies, the integration of religious texts and practices into digital formats requires sensitive handling to ensure respect and accuracy. Developers often engage with religious leaders and communities to ensure that the digital representation of religious practices aligns with traditional interpretations and respects the sanctity of these practices (Foltz and Foltz, 2003). This interaction helps in mitigating risks associated with the potential misuse or misinterpretation of religious content when presented in a digital format.

Despite the advancements and innovative solutions applied in overcoming these development challenges, gaps in understanding and implementation persist. There is a continual need for research into managing platform fragmentation more effectively and optimising UI/UX designs to cater to increasingly diverse user bases. Additionally, the digital representation of religious practices still necessitates further study to ensure that these technologies are used appropriately and constructively within religious communities (Lowdermilk, 2013; Foltz and Foltz, 2003).

As mobile technology advances, the interplay between technological innovation and user-centric design becomes more intricate and interconnected. The future of mobile app development, particularly in fields that blend technological interfaces with user-centred and religious frameworks, promises further challenges but also greater opportunities for innovative solutions. The ongoing evolution in this field underscores the importance of adaptive strategies that are responsive to both technological advancements and the nuanced needs of users across different contexts. This dynamic field requires developers to not only keep pace with rapid technological changes but also deeply engage with the users they aim to serve, ensuring their needs and values are reflected in the technology created.

3.2 Methodology

Project Management

The development of the Northumbria ISOC app was meticulously orchestrated using the Agile project management framework to cater to the dynamic and fast-paced environment of university app development. The selection of Agile was strategic, emphasising its adaptability, which proved vital in addressing the rapidly changing needs of the user community. Key Agile principles such as flexibility in design, iterative development through successive refinements, and ongoing feedback from users were instrumental throughout the project. This collaborative effort was particularly evident in the working relationship with the ISOC committee at Northumbria University, which ensured that the app's features resonated with the expectations and real-world use cases of its intended users (for detailed requirements, see table 1 & 2 in appendix B).

The methodology enforced the idea of regular updates, allowing the development to swiftly act on feedback and to continuously enhance the app's functionality, ensuring it remained attuned to the

evolving preferences of the Northumbria Muslim community. This iterative process of development and refinement, a core component of Agile methodology, is visualised in Figure 3. The figure maps out the development journey starting from the initial research and idea generation to prototype development, followed by a feedback loop where ISOC committee and user feedback are integrated, leading to further iterations and refinements of the app.

Figure 3 demonstrates how initial research and competitive analysis provided the insights necessary for ideation and feature brainstorming. Prototypes were then crafted based on this research, setting the stage for a feedback loop with the ISOC committee and the general user base. This valuable feedback was instrumental in refining the app, as captured in the subsequent iterations outlined in the figure. As the final versions of features were reviewed in comparison with ongoing research and trends, further feedback, if any, was utilised for additional iterations and enhancements. Testing and validation were conducted as per the specific feature requirements, ensuring that each element of the app not only met but exceeded user expectations.

The Agile flow, depicted in Figure 3, allowed for an efficient transition to the next feature, ensuring a cohesive and user-focused development lifecycle. The adaptability facilitated by this approach also meant that future enhancements could be seamlessly integrated, maintaining the relevance of the app in the face of new trends or user requirements. By referring to Figure 3, stakeholders can gain a comprehensive understanding of the iterative and user-centred approach that was fundamental in the app's development. This visual representation underlines the Agile methodology's impact on the project, emphasising the critical nature of adaptability, user engagement, and feedback in the creation of digital solutions in an academic context.

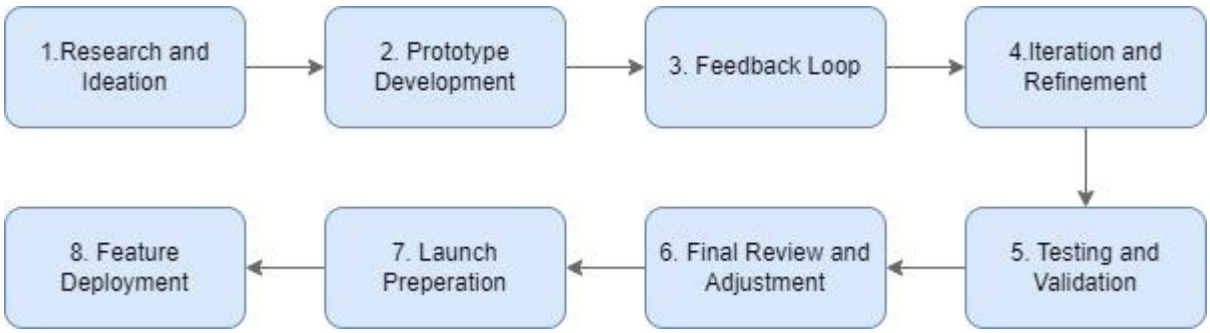


Figure 3. Agile Process Flow Diagram

Planning Phase

In the planning phase, a robust framework for the app's development was established through a deliberate and strategic application of Agile principles. This phase was marked by thoughtful feature prioritisation shaped by comprehensive consultations with the Islamic society committee and insights gleaned from potential users. The app's development was guided by the immediate needs of the Muslim student community at Northumbria University, prioritising essential functionalities such as prayer times, Quran access, and an events calendar.

There was a high priority placed on security (NFR1), choosing to integrate Firebase for resilient cloud services and secure user authentication (FR1). This decision was further reinforced by the adoption of token authentication using a Laravel backend, enhancing data integrity and alignment with contemporary app development standards.

The Agile methodology's monthly sprints focused sequentially on distinct functionalities, starting with the foundational homepage and methodically integrating more sophisticated features like the events calendar (FR4). The use of a Gantt chart was instrumental in managing the project timeline effectively, affording the flexibility necessary to incorporate continuous feedback and to reconcile

development efforts with academic responsibilities. This structured approach ensured a logical and systematic development of features, laying a solid foundation for the subsequent design and prototyping phases.

Refer to Appendix A for the detailed Gantt Chart, which illustrates the strategic planning and time management that underpinned the development process. This careful planning was pivotal in shaping an application that is both functional and reflective of the users it aims to serve.

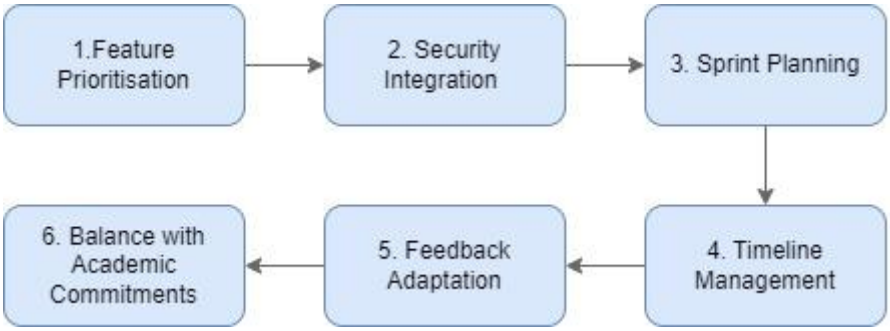


Figure 4. Planning for the Northumbria ISOC App

Design and Prototyping Phase

The initial concept for the Northumbria ISOC app was visualized through paper sketches, setting the foundational ideas that would evolve through rigorous user feedback and testing. This phase was crucial, allowing for early user input from the ISOC committee, ensuring the app's features met community needs and aligned culturally and religiously with users' values.

With digital prototyping tools like Figma, these concepts were refined into tangible prototypes, facilitating a collaborative design process. This phase emphasized enhancing user-centric functionalities, such as the digital Quran and interactive mosque visuals, and adhered to principles of consistency, navigation, and accessibility. The iterative cycle of development, testing, feedback, and refinement—depicted in Figure 5—ensured that the app was intuitive for a diverse user base and balanced technical feasibility with user needs. The final design featured a clear prayer timetable, a user-friendly Quran interface, and a comprehensive events section.

Throughout, Agile methodology enabled a responsive approach to the dynamic needs of the user base, allowing the app to evolve into a robust tool for individual spiritual practice and community engagement.

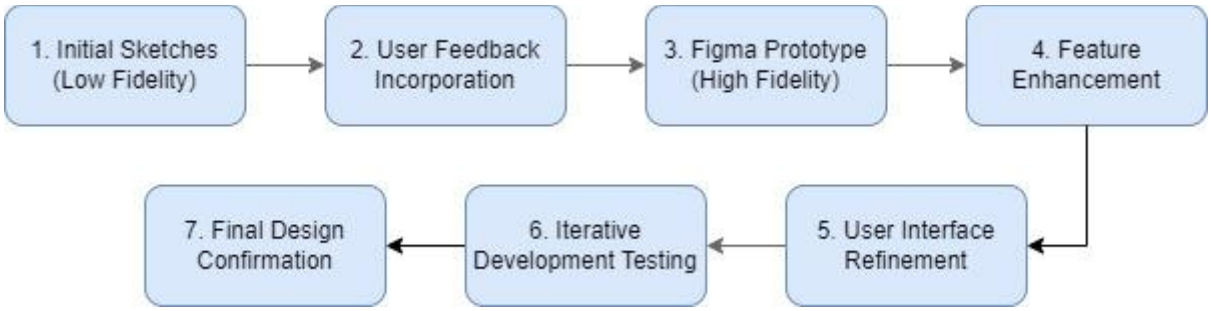


Figure 5. Design and Prototyping Workflow

Implementation Phase

The implementation phase marked a significant transition of the Northumbria ISOC app from concept to reality, as detailed in Figure 6. Employing React Native facilitated a fluid user interface and cross-

platform adaptability, while Laravel's robust backend framework ensured scalable and secure user interactions. Firebase integration provided secure user authentication, enhancing the app's scalability.

Key features such as precise prayer time management were implemented using a local SQLite database for real-time updates. The Quran feature was enhanced by transitioning to a custom API, improving navigation through religious texts. Athkar phrases were tailored to time-of-day, adding contextual value, and Google Maps integration enabled easy access to local mosque locations.

The systematic documentation of this phase reflects a commitment to user-centred design principles. Prior to final integration, each feature underwent rigorous testing to ensure flawless functionality. The deployment strategy remained flexible, managed by the Northumbria ISOC committee to align with their resources and schedules. This approach not only ensured operational readiness but also empowered the committee to effectively manage and support the app post-launch.

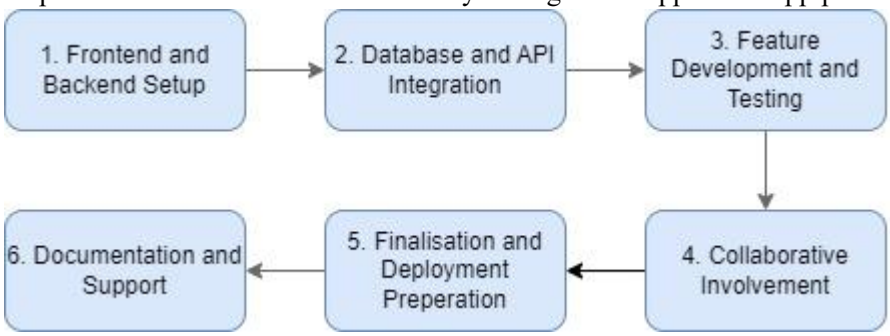


Figure 6. Implementation Process Flowchart

Testing and Evaluation Phase

The Testing and Evaluation Phase for the Northumbria ISOC app, as detailed in Figure 7, was structured to ensure the app's robust functionality and seamless user experience. The process began with Unit Testing, where each app component, like the Quran feature and event management functions, was individually assessed for proper functioning. This stage was crucial for isolating and resolving issues at the component level (NFR6).

Following this, Integration Testing was conducted to ensure that these components worked harmoniously together, confirming the system's integrity when operating. This testing was vital for verifying the seamless interaction between different parts of the app, such as integrating the prayer times with the events calendar.

The next critical step involved User Acceptance Testing (UAT), where a diverse group of users from the university's prayer facility, including both students and faculty, actively tested the app. They engaged in practical tasks such as setting up accounts and managing events, providing invaluable feedback. This feedback led to significant enhancements, notably the transformation of the Quran interface from a simple swipe mechanism to a more structured tabbed layout, thus improving navigability and user satisfaction (NFR6), as depicted in Figure 7.

Additional functionalities were integrated based on this user feedback, including detailed event management options, which aimed to boost the app's inclusivity and functionality. Technical challenges identified during UAT, such as content overflow in the Quran feature and timing issues in the prayer countdown, were swiftly addressed to ensure the app's reliability.

Performance Optimisation was also a focus, with metrics such as API load and response times closely monitored to enhance efficiency and reduce latency. The final phase of testing, as shown in Figure 7,

involved evaluating the app on various devices to confirm its cross-platform compatibility and readiness for potential release.

Although the app was not launched publicly, the rigorous testing within a controlled environment demonstrated its effectiveness in enhancing the religious and communal life of Muslim students and staff at Northumbria University. This phase validated the app's functionality and its capacity to effectively support community engagement and spiritual practices, ensuring that the Northumbria ISOC committee could manage and maintain the app effectively post-launch.

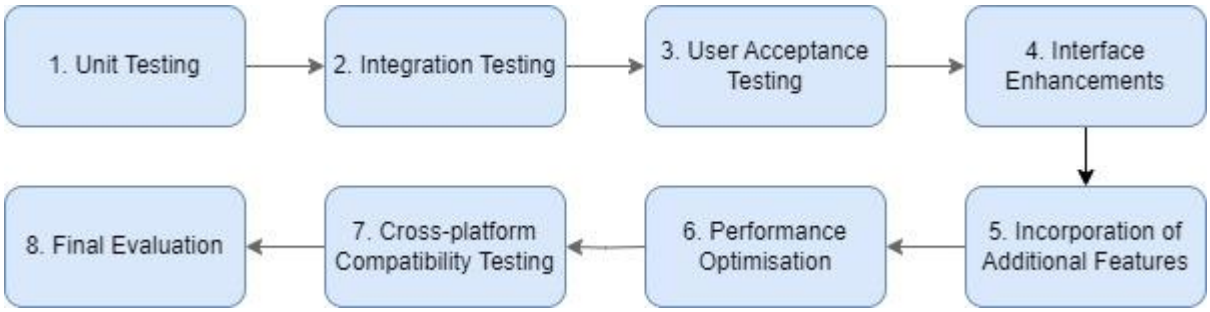


Figure 7. Testing and Evaluation Flowchart

4. Practical Work

This section details the development and testing of the Northumbria ISOC app, designed to enhance the university experience for Muslim students and faculty at Northumbria University. Integrating features like event coordination, accurate prayer times, and Quranic content, the app is tailored to meet the unique needs of its users. The development, guided by Agile methodologies, included continuous feedback loops and rigorous testing to refine functionality and user interface.

Key enhancements, such as the transformation of the Quran feature from a basic dropdown to a sophisticated tabbed interface, demonstrate the application's evolution through iterative design and user feedback. Security improvements through streamlined Firebase authentication processes also highlight significant developmental strides.

The practical work extends beyond development to include testing phases that ensured each app feature was both functional and user-friendly, setting a standard for multifunctional resources that support both academic and spiritual needs of Muslim students.

Design

The foundation of the Northumbria ISOC app's design lies in its carefully crafted architecture, which consists of three pivotal components: frontend development, backend development, and API integration.

Frontend Development

The Northumbria ISOC app leverages Expo for React Native development, serving as the interface through which users interact with its various features, including the events section and prayer times. Expo was chosen for its streamlined setup process and its ability to alleviate the burden of configurations, allowing developers to focus more on feature development.

The decision to utilise React Native for frontend development stemmed from its cross-platform capabilities, enabling the app to cater to both iOS and Android users without the need for separate codebases. This approach significantly expands the app's reach and ensures a consistent user experience across different devices. By opting for React Native over platform-specific languages like

Swift or Kotlin, the need to develop and maintain separate apps with identical functionalities is eliminated, streamlining the development process and maximising efficiency.

Throughout the frontend development phase, continuous feedback loops with members of the ISOC committee at Northumbria University and general app users highlighted a need for more efficient navigation within the Quran section. Originally featuring a simple dropdown menu, it was reported by users that scrolling through to find specific Surahs was time-consuming. In response, a tabbed interface was implemented, categorised by 'Surah', 'Page', and 'Juz'. This new design not only

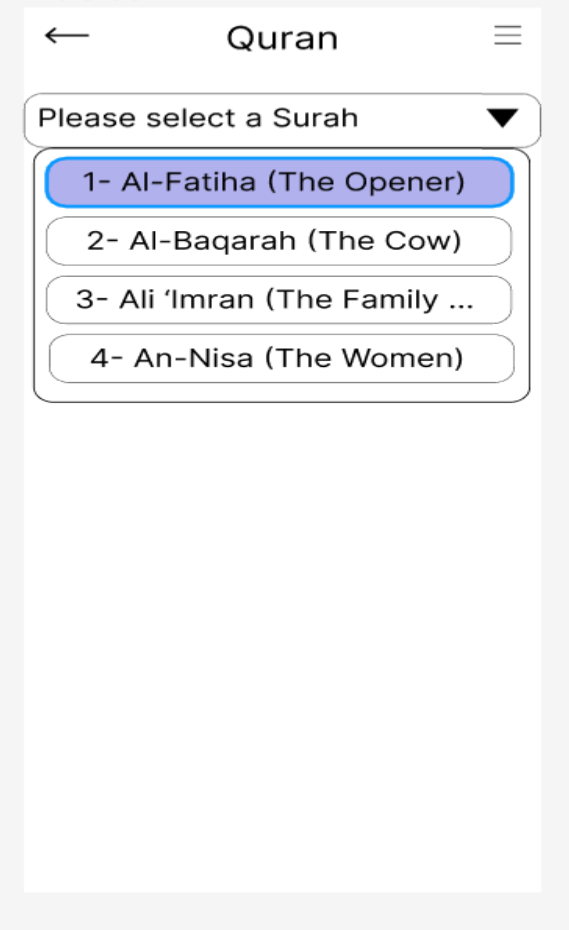


Figure 8. The original Quran section interface with a dropdown menu, designed for basic navigation but found to be time-consuming by users.

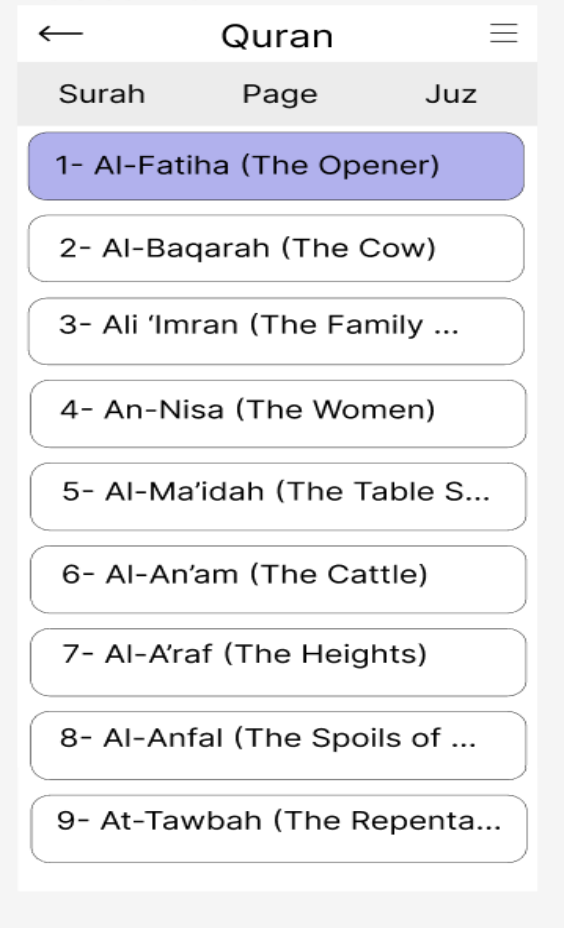


Figure 9. Updated interface incorporating tabs for Surah, Page, and Juz, introduced in response to user feedback for quicker and more intuitive navigation.

enhanced visual appeal but also significantly improved usability. The interface now allows users to navigate between sections with fewer interactions, making the app more accessible and user-friendly. This transition, vividly captured in Figures 8 and 9, illustrates the commitment to refining the app based on direct user insights.

This evolution in design not only demonstrates a responsive and adaptive development process but also aligns closely with the principles of user-centred design, ensuring that the app remains practical and appealing to its users.

Backend Development

In the development of the Northumbria ISOC app, Laravel (PHP) was selected as the primary framework due to its robust feature set and scalability, which are critical for supporting the extensive functionalities required by the app, including event management and real-time data processing for

daily prayer times. Laravel's eloquent ORM and expressive syntax simplify complex backend operations, making it an ideal choice for this project.

The decision to use SQLite as the database management system was driven by its lightweight architecture and ease of integration with Laravel. SQLite supports the application's need for a scalable, efficient data handling solution without the overhead associated with more complex systems. This choice was particularly pivotal given the necessity for the app to perform well on devices with limited resources, reflecting a design decision aligned with providing utility to a broad user base.

The database schema was meticulously crafted to optimise performance and data integrity. Each table was designed with meaningful names and columns that strictly relate to its function to prevent redundancy and ensure quick access. Special attention was dedicated to the format and storage of prayer times, a critical feature for the target audience. Data for each prayer time, such as Fajr, Sunrise, Dhuhr, Asr, and Maghrib, was stored in distinct columns formatted in HH:MM:SS to ensure consistency and precision. This arrangement not only facilitates efficient retrieval but also enhances the frontend's ability to render these times accurately, as demonstrated in Figure 10. Such structuring is essential for maintaining operational efficiency and user satisfaction, as it directly impacts the app's responsiveness and reliability.

	Date	Day	Fajr	Fajrlqamah	Sunrise	Dhuhr	Dhuhrlqa...	Asr	Asrlqamah
	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
1	01/01/2024	Monday	06:37:00	06:42:00	08:31:00	12:10:00	12:15:00	13:35:00	13:40:00
2	02/01/2024	Tuesday	06:37:00	06:42:00	08:31:00	12:11:00	12:16:00	13:36:00	13:41:00
3	03/01/2024	Wednesday	06:37:00	06:42:00	08:31:00	12:11:00	12:16:00	13:37:00	13:42:00
4	04/01/2024	Thursday	06:37:00	06:42:00	08:30:00	12:12:00	12:17:00	13:38:00	13:43:00
5	05/01/2024	Friday	06:36:00	06:41:00	08:30:00	12:12:00	12:17:00	13:39:00	13:44:00
6	06/01/2024	Saturday	06:36:00	06:41:00	08:29:00	12:13:00	12:18:00	13:40:00	13:45:00
7	07/01/2024	Sunday	06:36:00	06:41:00	08:29:00	12:13:00	12:18:00	13:41:00	13:46:00

Figure 10. Database Layout for Daily Prayer Times - Shows structured storage for prayer times, optimising data retrieval and consistency.

Furthermore, the backend utilises Laravel to develop RESTful APIs, fostering effective communication between the app's frontend and backend components. This integration is crucial for ensuring that data flows seamlessly across the application's architecture, enhancing user interaction and responsiveness to dynamic content such as prayer time updates or event modifications. For instance, the 'EventsController' handles event data interactions through methods like index, which retrieves all events, and store, which allows for the creation of new events. These methods utilise Laravel's routing and Eloquent ORM to efficiently manage HTTP requests and database operations, returning data in JSON format to ensure compatibility with modern web and mobile frontends, as shown in Figure 11.

```php
public function index()
{
    $events = Event::all();
    return response()->json($events);
}

public function store(Request $request)
{
    $event = Event::create($request->all());
    return response()->json($event, 201);
}
```

Figure 11. Laravel EventsController - Demonstrates RESTful API methods for managing event data such as creating events (store function) and retrieving events (index function).

API Integration

The Northumbria ISOC app relies on a variety of APIs to facilitate seamless communication between the backend and frontend components. Among these, RESTful APIs stand out as pivotal elements designed not only to enhance development efficiency but also to ensure stateless communication, a cornerstone of modern web architecture.

One significant advantage of leveraging RESTful APIs lies in their stateless nature, wherein each client request encapsulates all necessary information for server-side processing. This eliminates the need to maintain client state between requests, thereby streamlining the communication process and enhancing overall performance. By adhering to RESTful principles, the app achieves a more efficient and responsive user experience.

Moreover, API integration plays a crucial role in fortifying the app's security posture. Unlike direct database access, which poses inherent security risks, APIs enforce strict authorisation mechanisms to safeguard sensitive data. For example, the implementation of UID (unique user ID) authorisation ensures that only authenticated users, particularly those with administrative privileges, can access specific functionalities within the application. This granular access control helps mitigate potential security vulnerabilities and protects user credentials from unauthorised access.

Furthermore, API integration fosters collaboration between backend and frontend components, enabling seamless data exchange and facilitating dynamic content updates. For instance, the app's frontend can efficiently retrieve real-time data, such as event updates or prayer time modifications, through well-defined API endpoints. This harmonious interaction not only enhances user engagement but also contributes to the app's overall performance and reliability.

To better illustrate the operational flow of these interactions, a simplified flowchart is presented in Figure 12. This flowchart demonstrates the basic pathway through which user roles determine the level of access and interaction with the app's functionalities. By depicting how RESTful API calls are made based on user actions, Figure 12 provides a clear visual representation of the app’s handling of administrative and non-administrative roles. It intentionally omits complex error handling and security verification steps to maintain focus on the core process, thus serving as an educational tool for understanding the foundational API interactions within the app.

In essence, by embracing API integration, the Northumbria ISOC app achieves more than just communication between its backend and frontend; it establishes a robust foundation for security, efficiency, and user-centric design. This holistic approach ensures that the app delivers a seamless and user-friendly experience, thereby meeting the diverse needs of its users effectively.

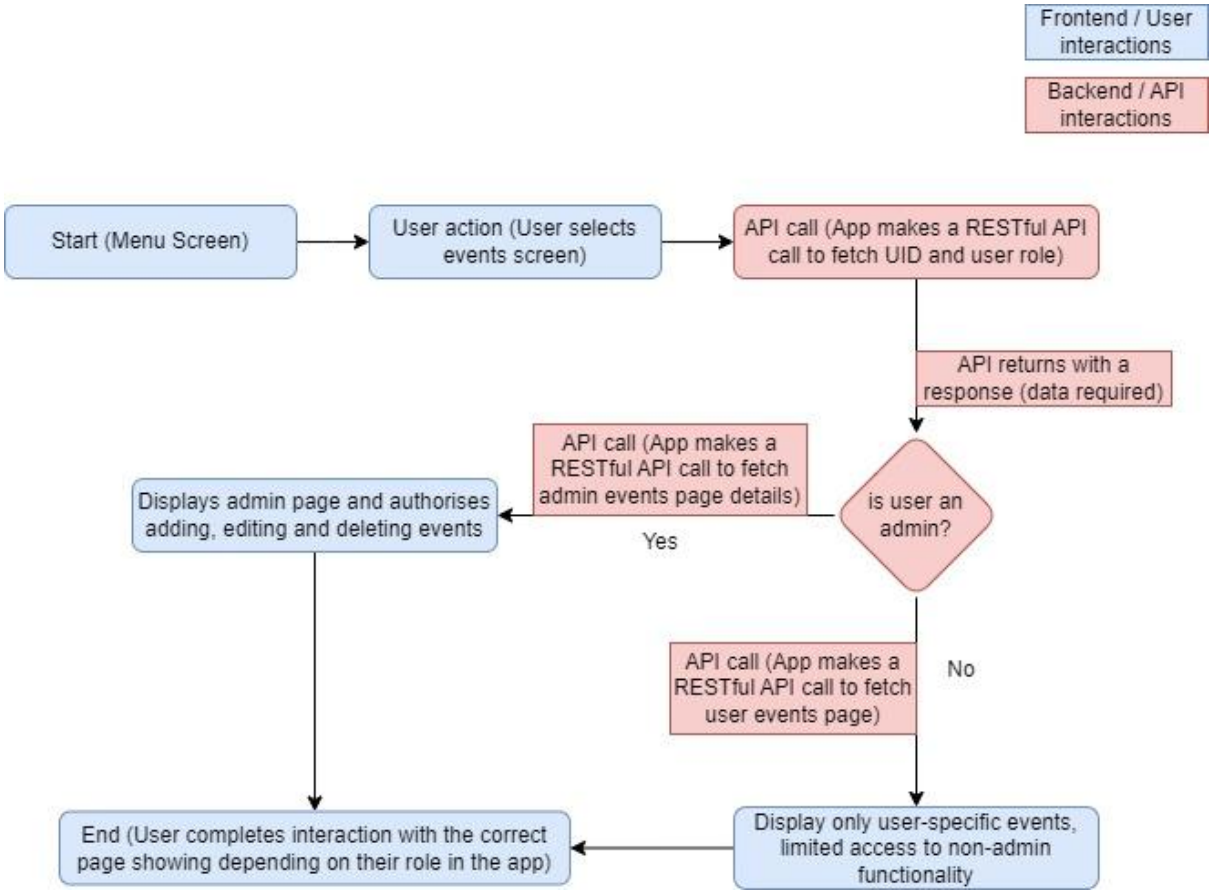


Figure 12. Simplified Flowchart Illustrating API-Driven User Role Authentication and Access Control for Event Management in the Northumbria ISOC App

Design Model

During the development process of the Northumbria ISOC app, Agile methodology served as the guiding framework, facilitating iterative development cycles and close collaboration between the developer and ISOC committee members. This iterative approach enabled continuous refinement of the app's features and user interface to better align with user needs and preferences, echoing the principles outlined earlier in the design and prototyping phase (refer to Figure 5).

An illustrative example of Agile's impact on the app's development involves the refinement of the authentication mechanism based on user feedback. Initially, the authentication process relied on email verification, requiring the developer to confirm a user's admin status via email. Feedback from ISOC committee members highlighted the inefficiency and inconvenience of this approach, particularly when managing a large user base. As demonstrated in the design and prototyping phase, feedback is a critical component of the Agile process, shaping the evolution of the app's features to meet user expectations.

In response to this feedback, iterations on the authentication mechanism were made, seeking to streamline the process while maintaining security measures. The result was a user-friendly solution that allows admins to search for users by their unique usernames within the app interface and directly

assign or revoke admin privileges. This iterative refinement not only simplified the admin role management process but also reduced administrative overhead, enhancing the overall user experience.

The example of refining the authentication mechanism underscores the iterative nature of Agile methodology, wherein frequent feedback loops drive continuous improvement and refinement. By prioritising user input and adapting the authentication process accordingly, this ensured that the app's functionality closely aligned with the needs and preferences of ISOC committee members.

In conclusion, Agile methodology played a pivotal role in shaping the Northumbria ISOC app, enabling iterative development cycles and user-centric design decisions. By embracing Agile principles and responding to user feedback, as initially visualised in Figure 5 during the design and prototyping phase, a more intuitive and user-friendly app experience was created, ultimately enhancing user satisfaction and adoption.

Prototyping

The design evolution of the Northumbria ISOC app was driven by a structured prototyping process. Beginning with initial paper sketches, the developer outlined the app's layout and structure. Transitioning to Figma, interactive high-fidelity prototypes were developed, allowing for detailed visualisation and iterative refinement.

Feedback from ISOC committee members and users guided adjustments, ensuring alignment with user needs. Prototypes were iteratively refined before transitioning to app prototypes, providing a tangible representation of the final product. The prototyping process fostered efficiency, flexibility, and user-centric design.

The iterative development process for the Northumbria ISOC app involved thorough prototyping of core features like events, prayer times, and the Quran section. Each feature underwent a comprehensive prototyping phase, followed by review and feedback from the committee and users.

Initially, the prototype (Figure 13) included a Quranic recitation feature aimed at enhancing user engagement. However, feedback revealed that users found it unnecessary or distracting. Consequently, the feature was removed in the revised prototype (Figure 14) to better align with user preferences.

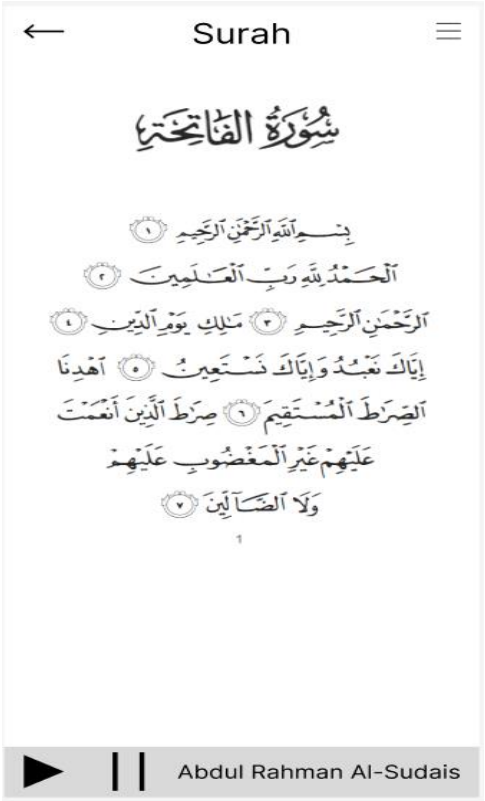


Figure 13. Initial Prototype with Quranic Recitation Feature

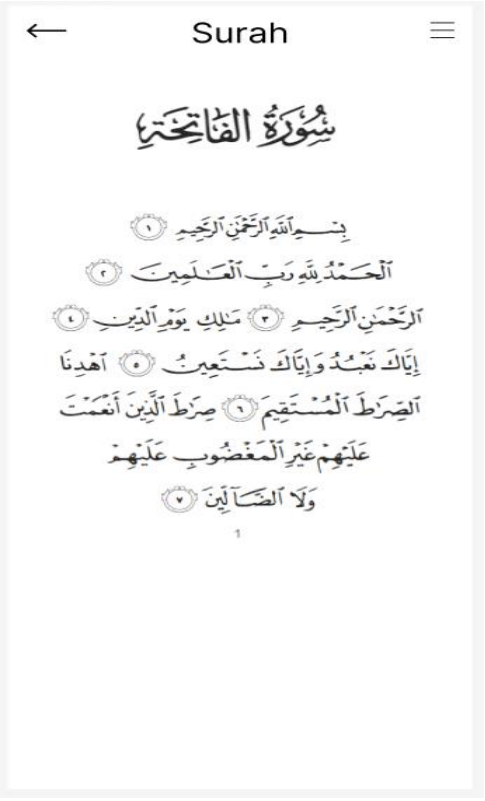


Figure 14. Revised Prototype without Quranic Recitation Feature

During user acceptance testing, one participant shared their thoughts on the high-fidelity prototypes, stating, "Experiencing the high-fidelity prototypes was like getting a sneak peek into the future of our app. With swiping gestures and countdown timers, using it felt natural and intuitive. Our input wasn't just heard; it was embraced, making the final product a true reflection of what we, as users, truly need and appreciate."

The transition to high-fidelity prototypes introduced more interactive elements, such as buttons and navigation gestures, to provide a realistic preview of the final product. Notably, enhancements were made to the home page, where features like prayer times were visually improved by highlighting upcoming prayers and adding countdown timers for quick reference.

For the Quran feature, swiping gestures were implemented for seamless navigation between tabs, chapters, and pages, enriching the user experience. Feedback from the ISOC committee during this phase allowed for final adjustments to accommodate user preferences.

While the events feature initially met requirements, adjustments were necessary for admin functionality verification, leading to slight modifications in the signup screen where the initial idea of users toggling a switch to sign up as admins changed to admins assigning other admins using the admin settings.

The addition of a chatbot, while not a priority for the committee, was deemed beneficial based on previous research. High-fidelity prototypes facilitated the identification of logic flaws and provided stakeholders with a clearer visualisation of the app's functionality, enabling quick adjustments and ensuring alignment with user needs.

UI Design Principles and Iterations

The user interface (UI) design holds paramount importance in the Northumbria ISOC app, serving as the primary point of interaction for users with its various features. Key design principles, including accessibility, navigability, and consistency, underpin the mobile application's UI design. A user-friendly interface that ensures ease of navigation and consistency across features is crucial for engaging users effectively. Accessibility is equally vital, fostering an inclusive environment where users from diverse demographics can utilise the app without encountering barriers or requiring additional assistance.

In crafting the UI, user needs and feedback are prioritised, alongside considerations for technical implementation. This approach ensures the development of an application tailored to the specific requirements of its users. However, striking a balance between user needs and technical feasibility, as well as addressing the committee's specific requirements, is essential for the app's success.

Mobile application users greatly benefit from intuitive and easy-to-navigate interfaces, as complex or unclear features can deter usage and even lead to app uninstallation. The Northumbria ISOC app prioritises intuitive design and accessibility, ensuring all features are easily accessible and straightforward to use. For instance, commonly recognised UI components such as the three-line menu button and back arrow are employed for seamless navigation (see Figure 15 and 16). These familiar elements enhance user experience by providing clear navigation cues.

Moreover, the app's menu features descriptive yet concise labels, such as "Quran" and "Events," to guide users effectively. This thoughtful approach reflects a user-centric design ethos aimed at improving overall usability. Additionally, the app's colour scheme, comprising cream, white, black, and blue tones, was selected for its appropriateness in an Islamic context. Research suggests that such colours, particularly cream, can enhance readability and ease navigation, especially for users with dyslexia, compared to colours like red and yellow which may disrupt reading (see Figure 15 and 16).

The inclusion of Figures 15 and 16 illustrates how these design considerations manifest in the app's interface, showcasing the use of common UI elements and the simplicity of navigation.

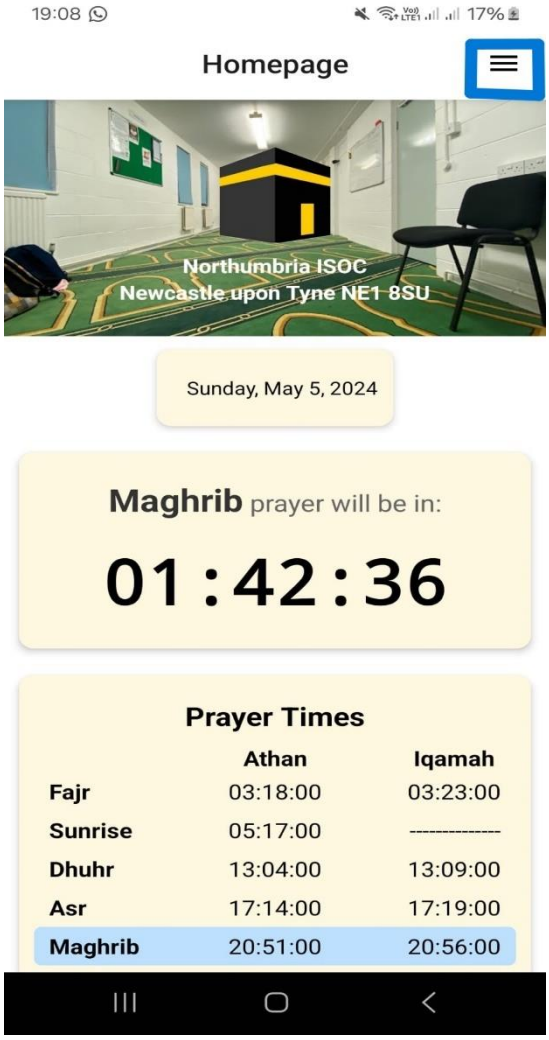


Figure 15. Showcases simple navigation elements such as the three-line menu button, enhancing user access to prayer times and other features.

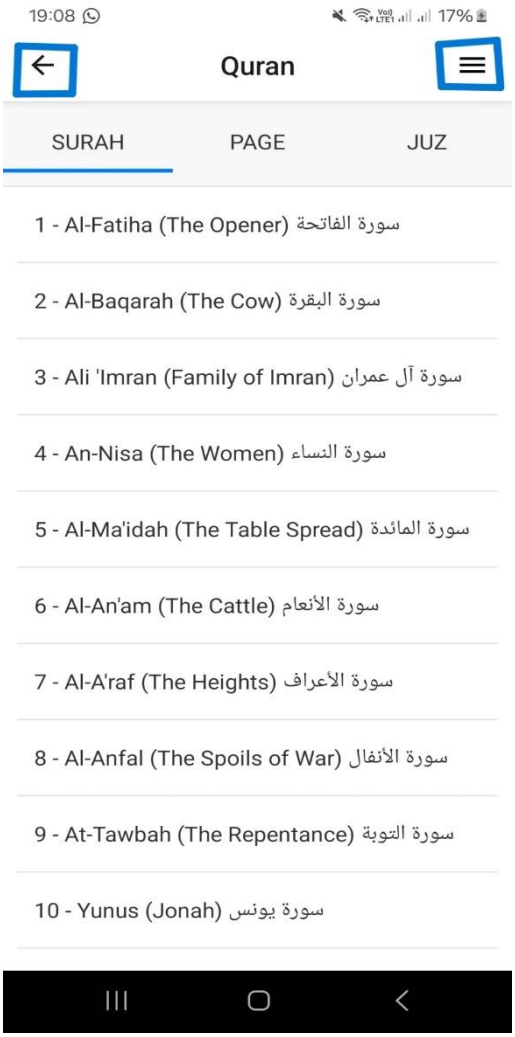


Figure 16. Features an easy-to-navigate index of the Quran, utilising familiar back arrow and page tabs for seamless user interaction.

The Northumbria ISOC app caters to a diverse user base, including individuals with varying levels of digital literacy, such as faculty members and students. Recognising this diversity, the app's design prioritises accessibility and usability to ensure all users can engage with its features effectively.

To accommodate users with less digital literacy, the app presents information in a clear and easy-to-read manner, employing sufficient contrast to differentiate interactive elements like buttons, which are highlighted with a blue background for enhanced visibility. This approach not only aids users in locating interactive features but also ensures that important information stands out amidst the app's content.

Moreover, the emphasis on intuitive navigation throughout the app further supports users of all proficiency levels in accessing its functionalities with ease. By maintaining consistency in design and layout, the app fosters familiarity and confidence among users, regardless of their digital proficiency.

Figures 17 and 18 provide visual representations of these principles in action, showcasing how the app maintains consistency in formatting data and provides clear visual cues to distinguish interactive

elements from static text. These examples highlight the app's commitment to delivering a user-friendly interface that accommodates the needs of all users.

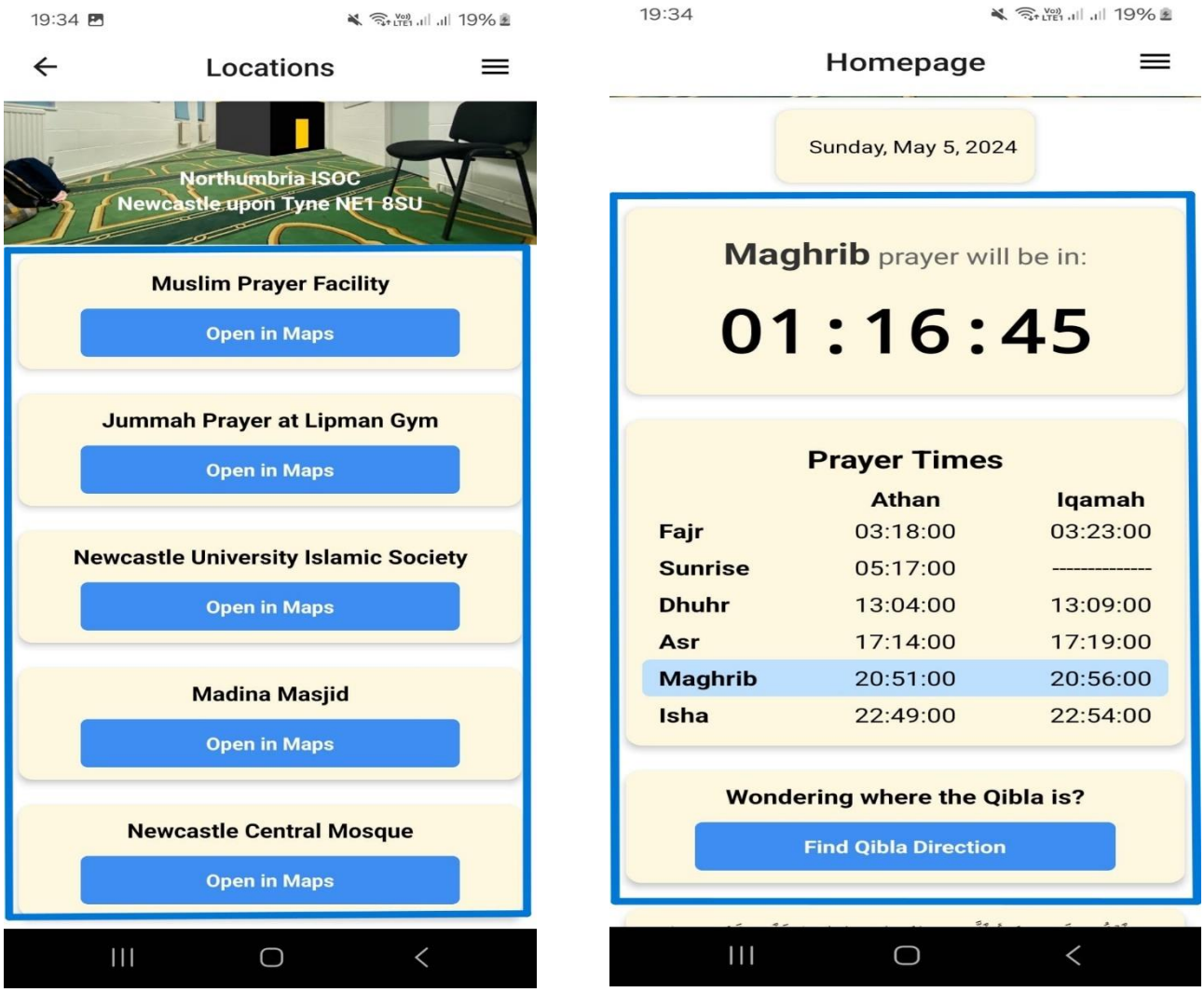


Figure 17 & 18. Consistent formatting in displaying data and differentiating between interactive elements and other data using similar backgrounds.

The Northumbria ISOC app underwent thorough testing across various devices running both iOS and Android operating systems to guarantee a consistent user experience regardless of the device used. Elements within the app were meticulously designed to adapt to different screen dimensions and sizes, prioritising responsiveness over static elements. This approach ensures that all users, regardless of their device specifications, can seamlessly interact with the app's features.

Design Feature Justification

The decision to integrate features such as prayer times, digital Quran, calendar, and the 'Ask Me' feature into the Northumbria ISOC app was driven by a thorough understanding of user needs and industry trends. Recognising the limitations of existing communication channels like WhatsApp groups, the ISOC committee aimed to provide a consolidated platform for essential religious resources and information.

Conversations with faculty members highlighted a preference for an all-in-one app, prompting the inclusion of functionalities like event management and access to the Quran. Feedback from users and the ISOC committee emphasised the importance of accessing mosque locations, informing the app's development to address both religious and navigational needs.

The integration of these features reflects a user-centric approach, supported by research findings and industry best practices, to enhance the app's utility and user experience.

In the subsequent discussion, we will delve into the purpose of each feature and examine how they have been meticulously designed to meet the needs of our audience, along with the challenges encountered during their implementation and how they were effectively addressed.

Prayer Times

Central to the functionality of the Northumbria ISOC app, the Prayer Times feature offers Muslims a convenient means to access the timing of the next prayer. Positioned prominently on the homepage, this feature caters to the immediate needs of users, recognising the significance of prayer in daily religious practice. Studies, such as research conducted by (Laird et al. 2024) on pray.com, underscore the pivotal role of daily prayer features, with a staggering 87.2% of users engaging with this aspect. Informed by competitor analysis and user preferences, the final design presents a clear countdown to the upcoming prayer, ensuring users can effectively plan their day around their religious obligations.

The comprehensive display of prayer times for the entire day addresses the varied schedules of users, allowing them to manage their time effectively even during busy periods, such as exams or job interviews. With five daily prayers each occurring at distinct times, reliance solely on memorisation or disparate online sources for accurate prayer times proves impractical. Therefore, the app serves as a reliable resource, consolidating all prayer times for easy access, particularly in the context of Newcastle's diverse Muslim community.

However, the implementation of this feature posed challenges, particularly in sourcing accurate and reliable prayer times for the entire year. To mitigate potential confusion among users, the decision was made to integrate local mosque timings, ensuring authenticity and alignment with community practices. This endeavour involved manual compilation of prayer times into a comprehensive database, necessitating considerable time and effort to ensure accuracy and reliability.

Quran

Integrating the Quran into the Northumbria ISOC app serves as a pivotal feature, allowing the Muslim community to stay connected with their religious and spiritual aspects throughout their daily lives. This inclusion aligns perfectly with the app's overarching goal of providing a digital solution tailored to the needs of the Muslim community at Northumbria University.

Initially, the Quran feature was conceived as a simple dropdown menu for accessing the text. However, through rigorous user testing and feedback sessions, it became evident that enhancing the interface would significantly improve user engagement. Consequently, the UI was transformed into a more enriched tabbed interface, offering users the flexibility to explore the Quran by chapter/surah, page, or juz. This change not only increased interactivity but also provided users with multiple pathways to access and engage with the Quranic content, resulting in a more enriching user experience.

Despite its benefits, transitioning to the tabbed interface posed technical challenges, necessitating the creation of separate APIs to ensure accurate and efficient retrieval of Quranic data for each tab. This process was time-consuming but essential to ensure the integrity and authenticity of the Quranic content sourced from Quran.com. By meticulously organising the Quranic text into a structured database, data retrieval from backend to frontend was optimised, ensuring seamless access to the Quran for app users.

Incorporating user feedback and testing throughout the design and implementation process played a pivotal role in shaping the Quran feature's evolution within the app. This user-centric approach not only validated the decision to enhance the interface but also underscored the app's commitment to meeting the diverse needs of its users.

Events Calendar

The calendar feature marks a pivotal addition to the Northumbria ISOC app, transitioning it into a more focused platform tailored for university students and faculty. This functionality not only facilitates socialisation but also fosters community engagement, particularly beneficial for international students and faculty members seeking opportunities to connect with others. By providing a centralised platform for discovering and managing events, Muslim students gain access to a diverse array of opportunities while empowering the ISOC committee at Northumbria University to efficiently oversee event management tasks.

During discussions with the committee, it became apparent that engagement levels with events were suboptimal, potentially attributed to the challenges users faced in locating and accessing event information. As one committee member aptly noted, "We noticed a gap in engagement with our events, and it seemed like many users struggled to find all the events we organised."

Addressing this challenge required robust measures to verify admin privileges and secure user data transmission. Secure APIs were developed to facilitate seamless interaction while ensuring that only authorised personnel could manage events. The user interface was meticulously designed to streamline event management tasks, with distinct buttons for adding, editing, and deleting events. Utilising red colour accents for the delete button not only enhances navigation but also intuitively communicates administrative actions.

Moreover, a user-friendly interface was crafted to simplify event discovery for non-administrative users, mitigating previous issues such as event fragmentation across disparate communication channels like WhatsApp chats.

In essence, the calendar feature represents a strategic enhancement aimed at bolstering community engagement and streamlining event management processes within the Northumbria ISOC app. Through collaborative efforts and user-centric design principles, this feature empowers users while reinforcing the app's commitment to facilitating meaningful connections and enriching experiences within the Muslim community at Northumbria University.

Implementation

The implementation phase of the Northumbria ISOC app marks the transformation of conceptual goals into tangible features, serving the Muslim community at Northumbria University with a comprehensive digital solution throughout their university journey. With a clear design plan delineating the functionality of each component, development efforts focused on coding various features to meet user needs effectively.

For frontend development, Expo was chosen for its compatibility with React Native, offering cross-platform capabilities and streamlined setup for efficient development. On the backend, Laravel provided a robust and scalable framework, complemented by a SQLite database to facilitate seamless data storage and API connections between frontend and backend systems. GitHub was utilised for version control, enabling the developer to keep track of changes, maintain a history of modifications, and manage the project's codebase effectively.

In ensuring the app's compatibility across different devices and operating systems, rigorous testing was conducted using various devices with different operating systems. This multi-device testing approach ensured that the app functions as expected across a range of devices, enhancing its accessibility and usability for users with diverse preferences and technological environments.

However, the development process was not without its challenges. Rigorous testing and adjustments were necessitated by user feedback and ISOC committee meetings, leading to iterative refinements in UI design, administration access, and feature functionality. While these adjustments may have slightly

prolonged development, they ultimately contributed to the creation of a more user-centric and comprehensive application.

This section will delve into the technology stack employed and rationale behind its selection over alternative technologies. Subsequently, it will detail the setup of the application from frontend to backend and the implementation of individual features, highlighting improvements and addressing challenges encountered during development. An overarching reflection on the implementation process will conclude this section before transitioning to a discussion on the rigorous testing conducted to ensure the app's alignment with user needs.

Technology Stack

In the development of the Northumbria ISOC app, a meticulous selection of technologies and frameworks underpinned its architecture and functionality. The technology stack comprised:

- **Frontend Development**:
    - React Native, in conjunction with Expo, constituted the primary framework for frontend development. Leveraging React Native's cross-platform capabilities and Expo's streamlined setup process, the app ensured a cohesive user experience across both iOS and Android platforms.
- **Backend Development**:
    - Laravel, a PHP framework renowned for its robust feature set and scalability, served as the cornerstone of backend development. This choice facilitated efficient data management and processing, critical for supporting the app's extensive functionalities. The integration of SQLite as the database management system provided a lightweight yet efficient solution for data storage and retrieval, aligning seamlessly with Laravel's architecture.
- **API Integration**:
    - RESTful APIs formed the backbone of communication between the frontend and backend components. By adhering to RESTful principles, the app achieved efficient data exchange and seamless interaction, contributing to enhanced user engagement and responsiveness.
- **Authentication**:
    - Firebase Authentication was employed to bolster the app's security infrastructure. This authentication mechanism ensured secure access to user accounts, safeguarding sensitive data and reinforcing user trust.
- **Version Control**:
    - GitHub served as the primary version control system, facilitating collaborative development and providing a comprehensive history of code changes. Through version control, the app maintained code integrity and streamlined the integration of new features and enhancements.

Feature Implementation

Prayer Times

In implementing the backend functionality for prayer times, the initial challenge was sourcing the required data in a suitable format. Despite unsuccessful attempts to find a ready-made prayer time schedule in SQLite format, an alternative solution emerged from accessing a paper calendar for the current year (2024) from a local mosque. To streamline data management, a CSV file was created using Microsoft Excel, containing all prayer times for 2024 with appropriate column headings and date/time formatting to ensure compatibility with frontend requirements (See Figure 19).

Date	Day	Fajr	FajrIqamal	Sunrise	Dhuhr	DhuhrIqam	Asr	AsrIqamah	Maghrib	MaghribIq	Isha	Ishalqamah
01/01/2024	Monday	06:37:00	06:42:00	08:31:00	12:10:00	12:15:00	13:35:00	13:40:00	15:49:00	15:54:00	17:44:00	17:49:00
02/01/2024	Tuesday	06:37:00	06:42:00	08:31:00	12:11:00	12:16:00	13:36:00	13:41:00	15:50:00	15:55:00	17:45:00	17:50:00
03/01/2024	Wednesda	06:37:00	06:42:00	08:31:00	12:11:00	12:16:00	13:37:00	13:42:00	15:51:00	15:56:00	17:46:00	17:51:00
04/01/2024	Thursday	06:37:00	06:42:00	08:30:00	12:12:00	12:17:00	13:38:00	13:43:00	15:53:00	15:58:00	17:47:00	17:52:00
05/01/2024	Friday	06:36:00	06:41:00	08:30:00	12:12:00	12:17:00	13:39:00	13:44:00	15:54:00	15:59:00	17:48:00	17:53:00

Figure 19. Section of the prayer times csv in the correct format with suitable heading names for retrieval using an API

With the prayer time data formatted and stored in the SQLite database using DB Browser, the next step was to develop an API endpoint to retrieve this data and transmit it to the frontend. The backend logic for fetching prayer times for the current day involved querying the database based on the formatted date. The code snippet provided illustrates how this was achieved using Laravel's Eloquent ORM to interact with the database and construct the response in JSON format (Refer to figure 20 & 21).

```
public function fetchPrayerTimesForToday($formattedDate) {
    $todayPrayerTimes = PrayerTime::where('Date', $formattedDate)->first(['Fajr', 'FajrIqamah', 'Sunrise', 'Dhuhr', 'DhuhrIqamah',
     'Asr', 'AsrIqamah', 'Maghrib', 'maghribIqamah', 'Isha', 'IshaIqamah']);

    if (!$todayPrayerTimes) {
        return null;
    }

    return $todayPrayerTimes->toArray();
}
```

Figure 20. function to fetch prayer times from the database with the correct format, converting all the data retrieved into an array for efficient sending through API to frontend.

```
    // Return the formatted prayer times as JSON
    return response()->json($formattedPrayerTimes);
} else {
    return response()->json(['message' => 'No prayer times for today.']);
}
```

Figure 21. Response is returned in JSON format which is a compatible way for the frontend to process the data as required.

The code snippet showcases the backend logic for fetching and formatting prayer times, ensuring the response adheres to the required structure for consumption by the frontend. Following successful implementation and testing of the API using tools like Postman, which provides insights into loading times and data validation (See Figure 22), confidence in the backends ability to deliver accurate prayer time data to the frontend was established.

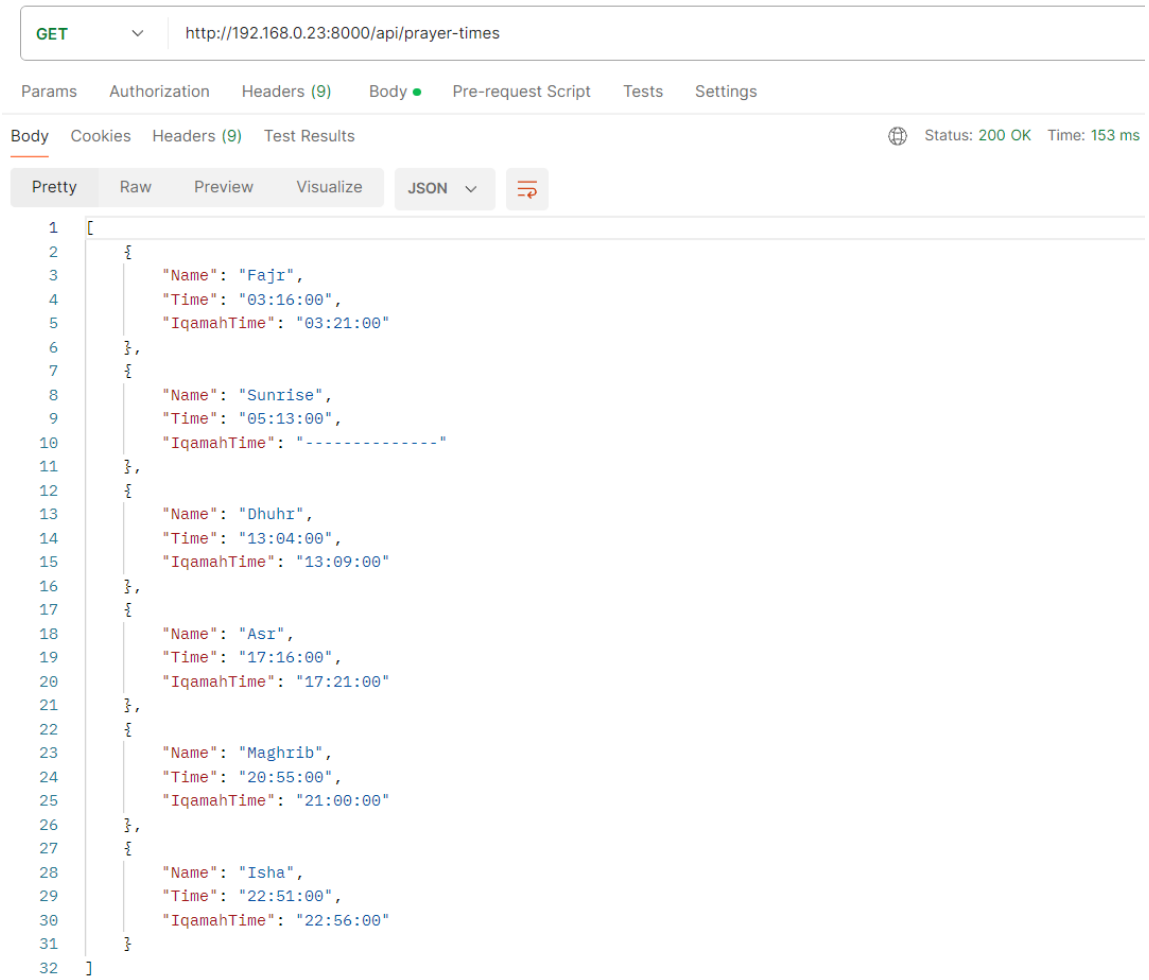


Figure 22. API endpoint tested using Postman shows a quick retrieval with a status code 200, along with all the data correctly formatted.

In the frontend implementation of prayer times, the decision was made to create a separate component for displaying prayer times on the homepage. This modular approach ensures that any modifications or updates to the prayer times component will not impact the entire homepage, enhancing maintainability and scalability. The prayer times component was designed to resemble a grid for easy readability, with a blue highlighter indicating the current prayer in real-time.

This code defines the 'PrayerTimes' component, which receives prayer time data and the index of the next prayer as props. It then renders a grid-like layout with prayer times displayed in rows, including the prayer name, Athan time, and Iqamah time. Styling is applied using React Native's 'StyleSheet' API to ensure a consistent and visually appealing presentation (See Figure 23).

```
const PrayerTimes: React.FC<Props> = ({ prayerTimes, nextPrayerIndex }) => {
  if (prayerTimes.length === 0) {
    return <Text>No prayer times available</Text>;
  }

  return (
    <View style={prayerStyles.prayerTimesContainer}>
      <Text style={prayerStyles.prayerTitle}>Prayer Times</Text>
      <View style={prayerStyles.prayerTimeHeaderRow}>
        <Text style={prayerStyles.headerName}></Text>
        <Text style={prayerStyles.headerTime}>Athan</Text>
        <Text style={prayerStyles.headerIqamahTime}>Iqamah </Text>
      </View>
      {prayerTimes.map((prayer, index) => (
        <View
          key={index}
          style={[
            prayerStyles.prayerTimeRow,
            index === nextPrayerIndex ? prayerStyles.highlighter : null,
          ]}
        >
          <Text style={prayerStyles.prayerName}>{prayer.Name}</Text>
          <Text style={prayerStyles.prayerTime}>{prayer.Time}</Text>
          <Text style={prayerStyles.iqamahTime}>{prayer.IqamahTime}</Text>
        </View>
      ))}
    </View>
  );
};
```

Figure 23. shows how the information is being sent through props in an ordered manner to then be styled and displayed on the frontend.

Overall, this frontend implementation encapsulates the prayer times functionality within a reusable component, facilitating easy integration into the homepage while maintaining a clear and intuitive user interface (See Figure 24).

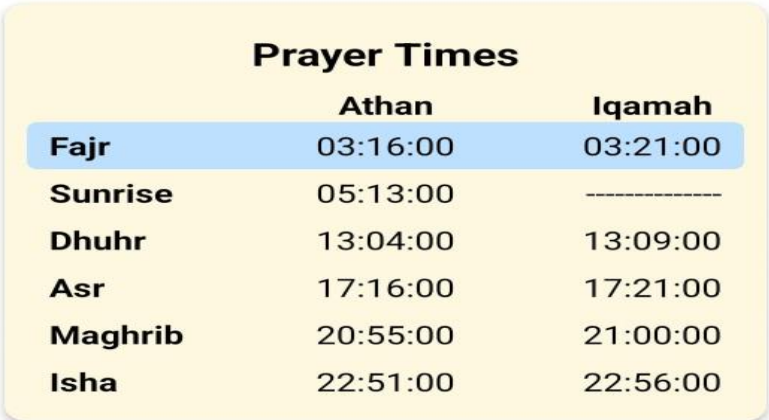


Figure 24. Clear representation of data that's easy to read, highlighter helps identify next prayer easily.

A countdown component was designed to display the time remaining until the next prayer. Here's a closer look at its functionality:

The component calculates the end time for the next prayer using the provided ‘nextPrayer’ prop. It adjusts this time to the Europe/London time zone for consistency across different time zones.

Subsequently, the component continuously updates the countdown timer every second, showing the hours, minutes, and seconds left until the next prayer (See Figure 25).

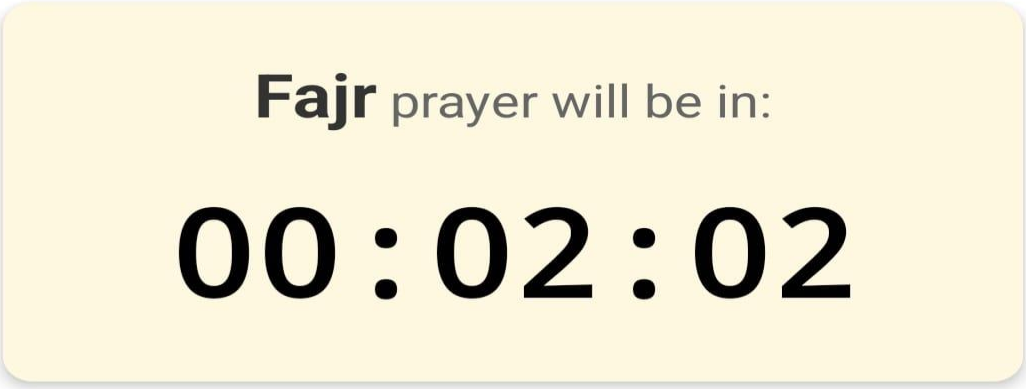


Figure 25. Large font to make it easy to read the next prayer time.

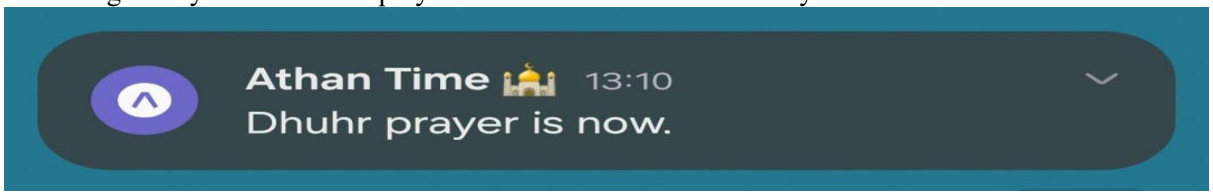
Moreover, the Countdown component facilitates push notification setup to alert users when the prayer time arrives by leveraging the Expo Notifications API. This method schedules a push notification with the target time as the trigger, utilising the Expo API which offers several benefits over traditional notification methods:

- **Reduction in Server Load**: Eliminates the need for continuous server-side processing to send notifications, reducing server load and potential failure points.
- **Simplified Development**: Expo provides a unified API for iOS and Android, simplifying the development process by eliminating the need to handle platform-specific notification services.
- **Scalability and Reliability**: Notifications are managed by client devices, allowing the system to scale seamlessly with the number of users without additional server resources.
- **Ease of Use**: Developers can easily set up and test notifications without the complexities of managing backend infrastructure for notification delivery.

This modern approach not only streamlines development but also enhances app reliability and improves the overall user experience.

To address challenges such as accurate time representation across time zones, the component integrates a reliable time zone package. This ensures precise countdown calculations regardless of the user's location.

Similarly, implementing push notifications traditionally involves complex configurations. However, by leveraging Expo's comprehensive suite of tools and APIs, including straightforward functions like ‘scheduleNotificationAsync’ (See Figure 26), the Countdown component streamlines this process, delivering timely reminders for prayer times with ease and efficiency.



```
const scheduleNotifications = async (targetTime) => {
  await Notifications.scheduleNotificationAsync({
    content: {
      title: "Athan Time 🕌",
      body: `${nextPrayer.NextPrayer} prayer is now.`,
    },
    trigger: targetTime,
  });
};
```

Figure 26. Using the 'scheduleNotificationAsync' function provided by expo to allow for easy to integrate push notifications, along with a title and body that dynamically change depending on the upcoming prayer.

Quran

The backend implementation of the Quran feature faced two primary challenges: sourcing authentic Quranic text and formatting it in a database conducive to easy retrieval and transmission. After extensive research, an SQL database containing Quranic verses was discovered. However, the formatting of this database was initially unsuitable for the desired application, as it presented each verse on a separate line, which did not align with the intended user experience of navigating Quranic pages like a book.

To address this challenge, inspiration was drawn from the structure of the quran.com web app, which offered an effective method for sourcing Quranic content into an SQLite database. The decision to adopt a page-by-page structure was based on usability studies indicating that users preferred a more traditional reading experience akin to physical Quran texts. This format facilitates a more immersive reading session, closely mimicking the flow of reading a printed Quran. A page-by-page structure was developed to ensure compatibility with the desired design, allowing users to swipe left and right to navigate between pages seamlessly. Additional columns were incorporated to enhance navigation flexibility, including 'surah_number' and 'surah_name' for organising Quranic verses by surah and 'juz_number', 'juz_start' and 'juz_name' for grouping verses by juz (See Figure 27).

surah_nu...	surah_name	ayah_number	surah_text	juz_number	juz_name	page	surah_start	juz_start	quran_...
1	Al-Fatiha (The Op...	7	Al-Fatiha (The O...	1	(الحمد لله)	1	1	1	1
2	Al-Baqarah (The ...	286	Al-Baqarah (The...	1	(الحمد لله)	2	2	1	2
2	Al-Baqarah (The ...	286	مْ إِنَّمَا نَحْنُ مُسْت...	1	(الحمد لله)	3	2	1	3
2	Al-Baqarah (The ...	286	بِنُۥٓ مِّمَّا نَزَّلْنَا عَلَىٰ...	1	(الحمد لله)	4	2	1	4
2	Al-Baqarah (The ...	286	هُوَ ٱلَّذِى خَلَقَ لَكُم ...	1	(الحمد لله)	5	2	1	5

Figure 27. Suitable headers for the Quran table, many columns to allow for a flexible approach for the frontend implementation.

Once the data was meticulously curated and verified, API endpoints were established to facilitate data retrieval in various formats. Four separate APIs were created to cater to different aspects of the Quran feature, including lists of surahs, juz, and pages, as well as a comprehensive endpoint for retrieving all Quranic pages. This endpoint featured a query mechanism to dynamically fetch pages based on user preferences, such as selecting a specific juz (See Figure 28).

```php
public function getPage(Request $request)
{
    $query = Quran::query();

    // Filter by surah if provided
    if ($request->has('surah_number')) {
        $query->where('surah_number', $request->surah_number);
    }

    // Filter by page if provided
    if ($request->has('page')) {
        $query->where('page', $request->page)->groupBy('page');
    }

    // Filter by juz if provided
    if ($request->has('juz_number')) {
        $query->where('juz_number', $request->juz_number);
    }

    $pageContent = $query->get();

    return response()->json($pageContent);
}
```

Figure 28. API that retrieves the whole Quran, featuring queries to allows for indexing whereby specific pages are called rather than the full Quran at once.

Thorough testing using tools like Postman ensured the correctness and reliability of the APIs, validating that data was retrieved and transmitted in the expected formats (See Figure 29 & 30). With the backend infrastructure in place, development of the frontend components could proceed confidently, building upon the solid foundation laid by the Quran APIs.

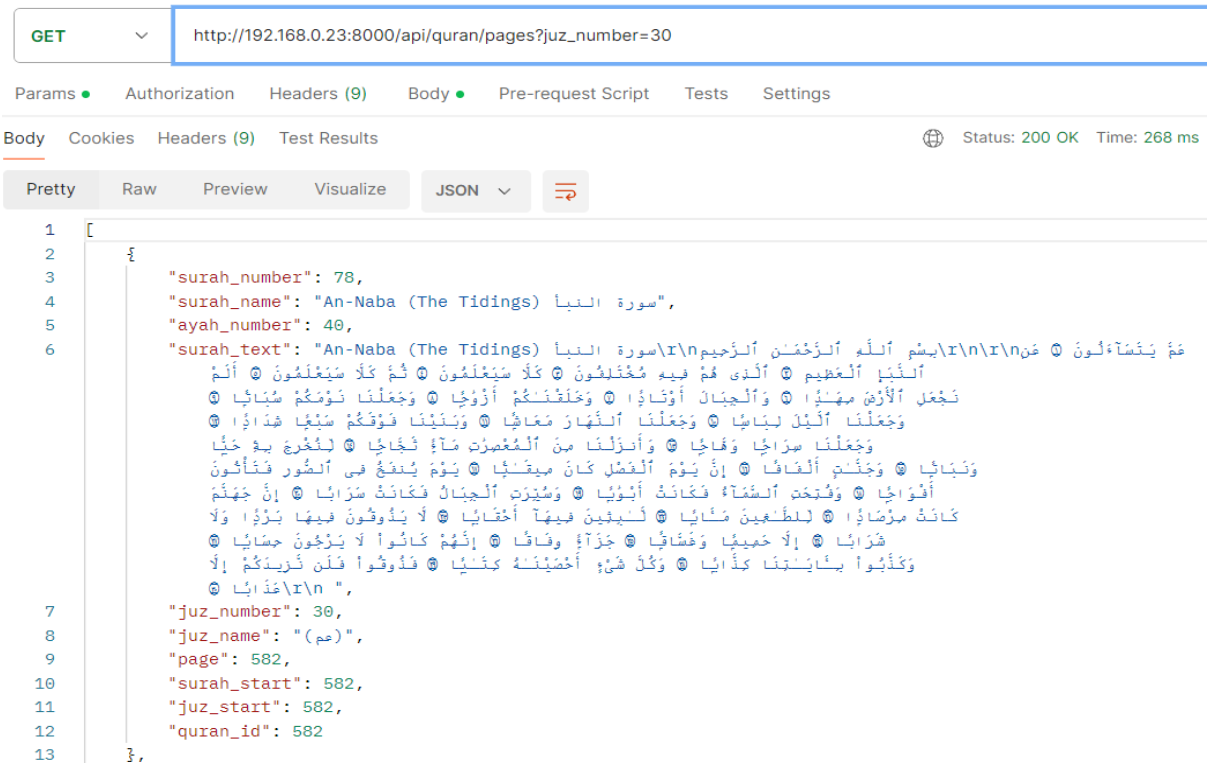


Figure 29. Example API call using the juz_number query which illustrates how indexing helps retrieve the start of the Juz allowing for more flexible navigation when it comes to building the frontend.

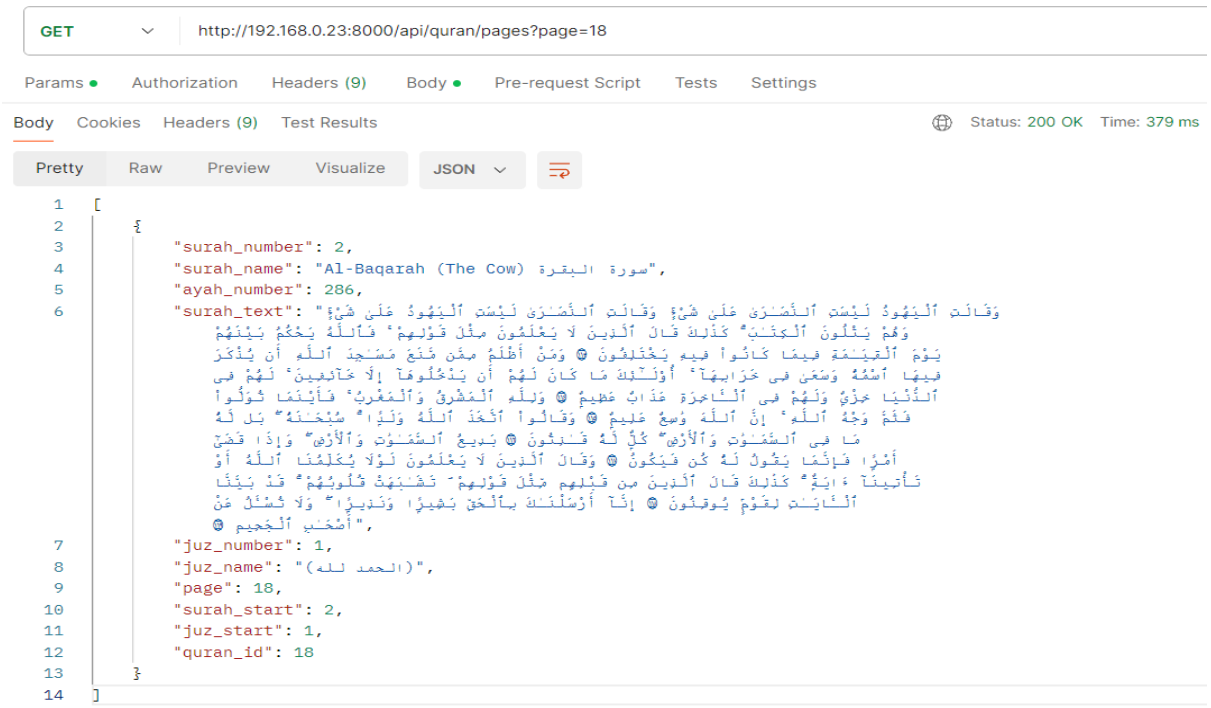


Figure 30. Example API call using the juz_number query which illustrates how indexing helps retrieve the start of the Juz allowing for more flexible navigation when it comes to building the frontend.

The frontend implementation of the Quran feature commenced with the setup of three fundamental tabs: Surah/Chapter, Page, and Juz. This decision arose from feedback provided by the ISOC committee, leading to a departure from the initial minimalist approach. Each tab serves as a distinct entry point for accessing Quranic content, providing users with intuitive navigation options (See Figure 31 & 32).

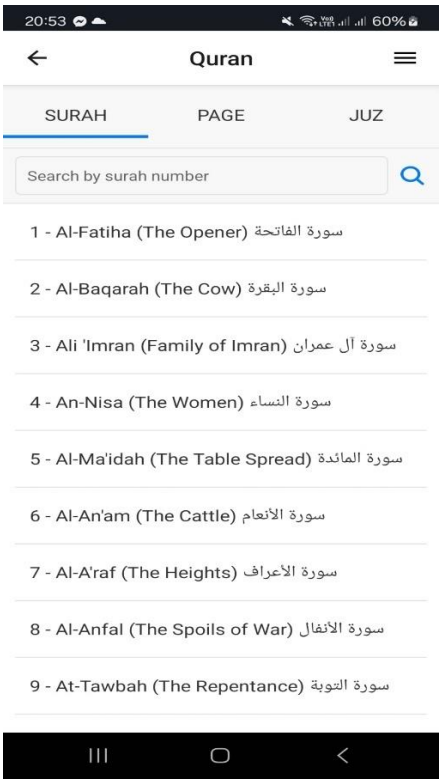


Figure 31. Tabbed interface showing a well organised section with surah/chapter names in English and Arabic

Once the tab structure was established, the focus shifted towards integrating the created API endpoints into the frontend. The Surah tab, being the most utilised format for reading the Quran, prioritised accessibility and usability. Leveraging the Surah API endpoint, a straightforward and intuitive scrollable tab was designed, listing all 114 Surahs/Chapters. Each surah is numbered and named in both English and Arabic, catering to diverse user demographics.

```
const tabs = ['surah', 'page', 'juz'];
const [selectedTabIndex, setSelectedTabIndex] = useState(0);

const handleTabChangeByIndex = (index) => {
  setSelectedTabIndex(index);
};

return (
  <GestureRecognizer
    onSwipeLeft={() => onSwipe('SWIPE_LEFT')}
    onSwipeRight={() => onSwipe('SWIPE_RIGHT')}
    config={config}
    style={styles.container}
  >
    <View style={styles.tabsContainer}>
      {tabs.map((tab, index) => (
        <TouchableOpacity
          key={index}
          style={[styles.tab, selectedTabIndex === index && styles.selectedTab]}
          onPress={() => handleTabChangeByIndex(index)}
        >
          <Text style={styles.tabText}>{tab.toUpperCase()}</Text>
        </TouchableOpacity>
      ))}
    </View>
  </GestureRecognizer>
);
```

Figure 32. Code related to setting up the tabs for the Quran feature.

Looking ahead, considering the diverse user demographics, integrating translations of the Quran into the app could significantly enhance accessibility and understanding for non-Arabic speaking users. This future enhancement would involve adding multilingual support to the existing framework, allowing users to toggle between Arabic and various translations as per their preference.

However, a minor issue surfaced when certain pages contained multiple Surahs. To address this, a pragmatic solution was devised, involving data redundancy within the database. By duplicating pages containing multiple Surahs and employing PHP's ‘groupBy’ method, the frontend seamlessly fetched and displayed Surahs, ensuring a consistent user experience.

Following the implementation of the Surah tab, attention turned towards the Page tab. Given that the Quran table was meticulously designed for page swiping functionality, integrating this tab was straightforward. Utilising a simple fetch request to retrieve data through the API, users can effortlessly navigate Quranic content by page number.

Similarly, the Juz tab mirrors the functionality of the Surah tab, albeit with a focus on Juz numbers and names. Leveraging the Juz list, this tab provides users with an intuitive means of accessing specific sections of the Quran, enhancing overall usability and accessibility.

As a final touch to enrich the user experience, the implementation incorporated the Toast library, inspired by competitor analysis of similar Quran apps. This library enables the display of animated alert messages, serving as visually appealing reminders for users. By seamlessly integrating these alerts, users receive timely notifications about their current Juz progress, facilitating a more immersive Quranic reading experience (See Figure 33).

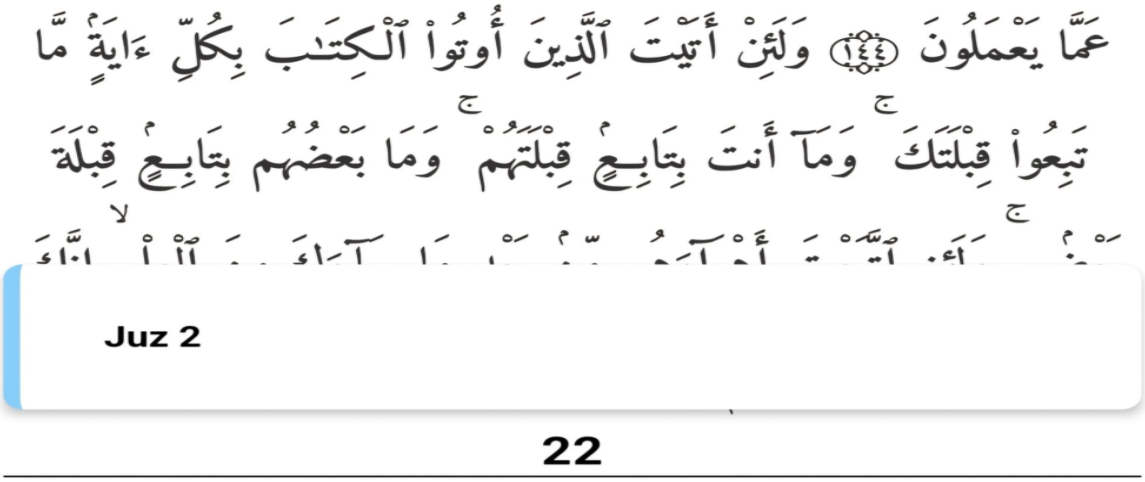


Figure 33. New Juz indication when a user starts reading a new Juz using the toast library.

Additionally, to further enhance user convenience and navigation, a search box feature was integrated into each tab. This functionality allows users to input the number of the Surah, Juz, or Page they wish to access directly, facilitating swift navigation to specific Quranic sections. Upon entering a number and initiating the search, the application validates the input to ensure it falls within the appropriate range for the selected tab. If the input is valid, the application navigates to the corresponding Quranic section. In case of an invalid input or out-of-range number, the user is prompted with an alert message, guiding them to enter a valid number within the specified range. By providing this search capability, users can quickly locate and navigate to their desired Quranic content, streamlining the reading experience and improving overall usability (See Figure 34 & 35).

```
    } else {
      alert(`Please enter a valid ${tabs[selectedTabIndex]} number between 1 and ${maxNumber}`);
    }
  } else {
    alert('Please enter a valid number');
  }
```

Figure 34. User alert when an invalid input is inserted, or no value is inserted.

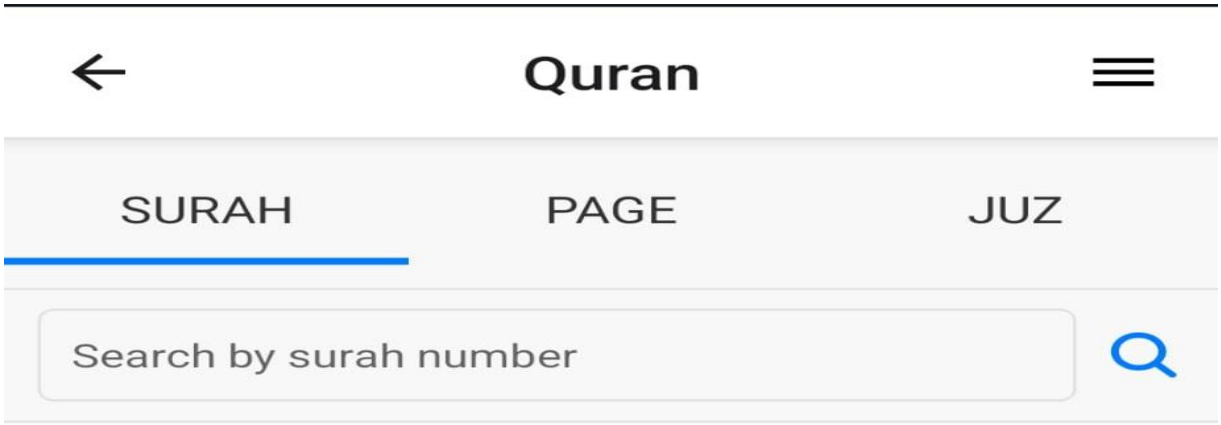


Figure 35. Search box to redirect users according to the tab they are on, and the number inserted in the query

Event Calendar

The events feature benefits from Laravel's powerful capabilities, making the setup of APIs straightforward and efficient. Leveraging Laravel's expressive syntax and built-in features, such as Eloquent ORM and resourceful routing, setting up CRUD (Create, Read, Update, Delete) operations for events becomes a seamless process.

```php
class EventsController extends Controller
{
    public function index()
    {
        $events = Event::all();
        return response()->json($events);
    }

    public function store(Request $request)
    {
        $event = Event::create($request->all());
        return response()->json($event, 201);
    }

    public function show($id)
    {
        $event = Event::findOrFail($id);
        return response()->json($event);
    }

    public function update(Request $request, $id)
    {
        $event = Event::findOrFail($id);
        $event->update($request->all());
        return response()->json($event);
    }

    public function destroy($id)
    {
        Event::destroy($id);
        return response()->json(null, 204);
    }
}
```

Figure 36. Events controller showcasing the different options that can be carried out from creating to editing to deleting events.

In the backend code provided, the ‘EventsController’ class showcases this simplicity. Each method corresponds to a specific CRUD operation: index for retrieving all events, store for creating a new event, show for retrieving a specific event by ID, update for modifying an existing event, and destroy for deleting an event (See Figure 36).

With Laravel's Eloquent ORM, database interactions are simplified, allowing for concise and readable code. For instance, the store method utilises the create method to instantiate a new Event model with the request data, while the update method efficiently updates the event attributes using the update method on the retrieved Event instance.

Furthermore, Laravel's robust routing system enables seamless endpoint configuration, ensuring that each API endpoint is neatly mapped to its corresponding controller method which was tested thoroughly using Postman. This enhances code organisation and readability, contributing to the overall maintainability of the project.

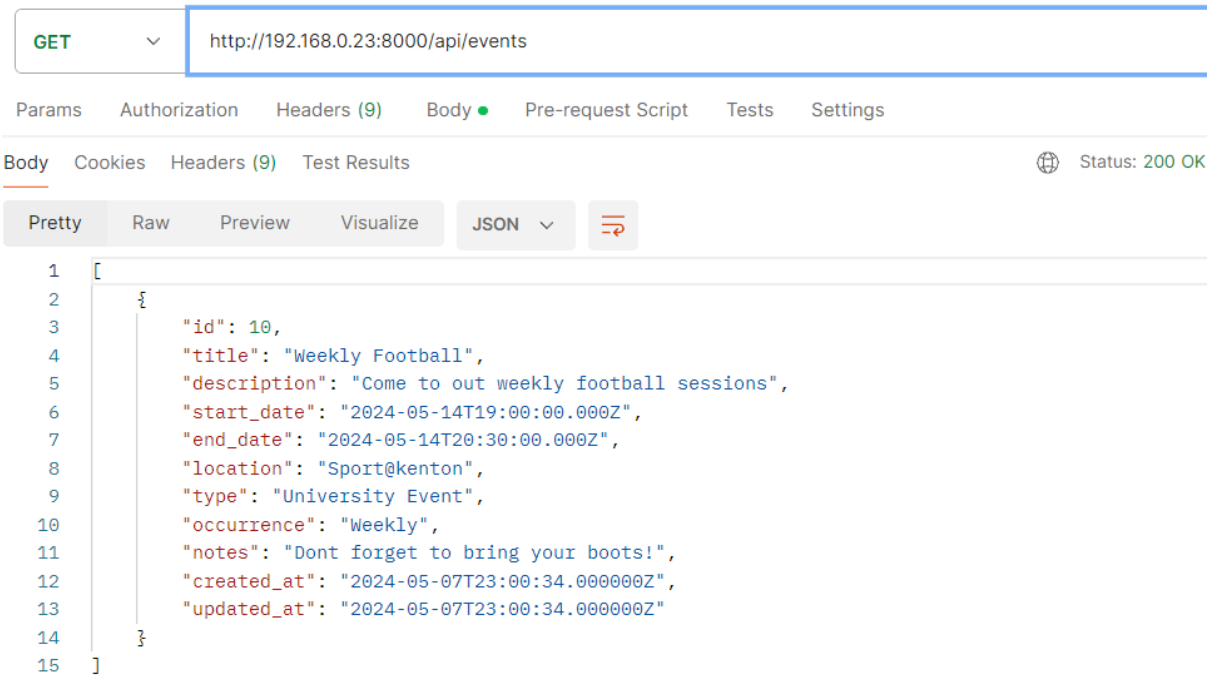


Figure 37. Events API that retrieves event data which can be used for both admins and regular users to view necessary information about upcoming events.

The frontend implementation of the event management system showcases a user-friendly interface that facilitates seamless interaction with event data. Utilising React Native, the frontend ensures a responsive and intuitive user experience, catering to both administrators and regular users.

For administrators, the interface offers clear and interactive elements, such as an add button for creating new events and edit/delete options for managing existing ones. This approach provides administrators with a streamlined workflow, allowing them to efficiently create, edit, and delete events directly from the application. Additionally, the interface displays event details

comprehensively, including title, date, location, type, occurrence, and notes, enhancing the overall management experience (See Figure 38 & 39).

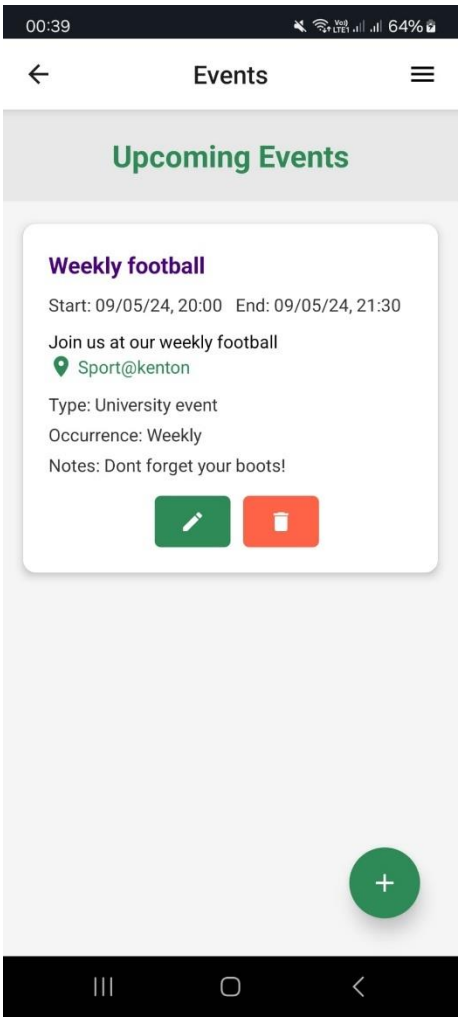


Figure 38. Admins interface showing clear buttons to manage events from adding to editing to deleting to simply viewing events.

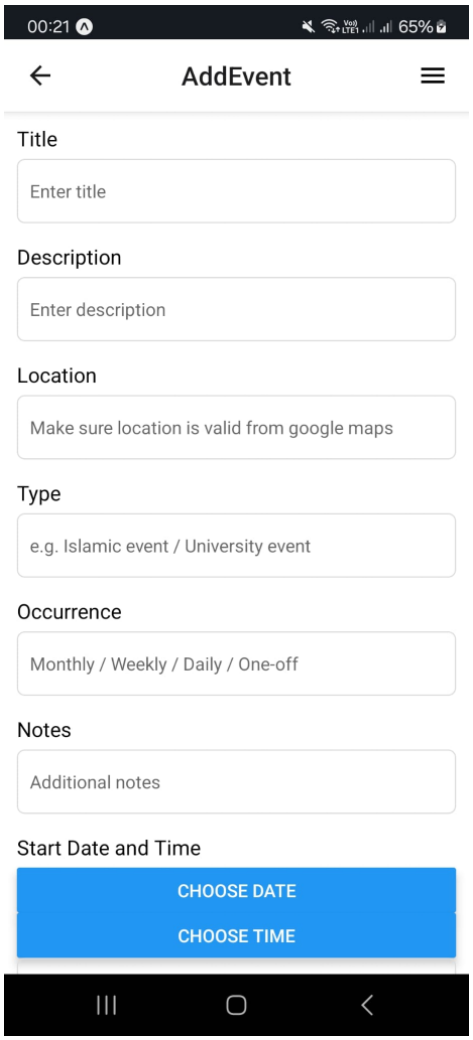


Figure 39. Adding form for admins for all event details

Compared to alternative frontend frameworks or libraries, such as traditional web-based solutions, React Native offers distinct advantages in terms of mobile application development. React Native's component-based architecture enables developers to build reusable and modular UI components, promoting code reusability and maintainability. This approach fosters a more structured and organised codebase, facilitating easier management and extension of the frontend application over time.

Furthermore, React Native's cross-platform compatibility allows for the development of mobile applications that run seamlessly on both iOS and Android devices, eliminating the need for separate codebases. This approach not only reduces development time and effort but also ensures a consistent user experience across different platforms.

Overall, the frontend implementation of the event management system exemplifies the benefits of leveraging React Native for mobile application development. By combining intuitive design principles with the flexibility and performance of React Native, the frontend offers a compelling solution for managing events effectively, catering to the diverse needs of administrators and users alike (See Figure 40).

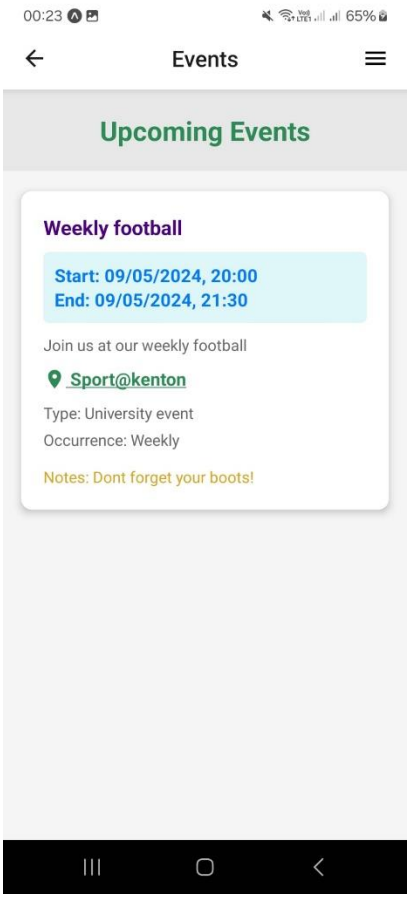


Figure 40. Regular user interface for viewing event details in a visually appealing manner, clearly distinguished times using styling, interactive location that redirects to google maps for a more specific map location.

The decision to utilise React Native for the frontend development of the event management system was strategically informed by its successful implementation in industry-leading applications such as Facebook and Uber Eats. These platforms have demonstrated how React Native can effectively enhance real-time user interactions and improve the efficiency of event-driven processes, crucial elements for managing events. React Native's proven capability to handle dynamic updates and seamless backend integration ensures that the application can manage complex event logistics and user interactions smoothly. This approach not only delivers a high degree of responsiveness and user satisfaction but also supports specific requirements of an event management system such as real-time updates, notifications, and interactive elements. The adoption of this industry-validated framework enables the project to benefit from its robust ecosystem and active community, ensuring that the application remains at the forefront of mobile technology advancements and capable of incorporating the latest features.

Testing

Testing Plan

The development of the Northumbria ISOC app was supported by a meticulous and multifaceted testing strategy, designed to ensure the application was robust, user-friendly, and met all functional requirements. Testing was integral throughout the app's development lifecycle, incorporating various testing methodologies to validate every component of the software.

- **Unit testing:** Unit tests were rigorously applied to critical backend functionalities, such as fetching prayer times, to ensure individual components operated as expected. Leveraging Laravel's built-in testing capabilities, these tests simulated functionality in isolated

environments to verify the correctness of the application logic without external dependencies. For example, tests ensured that the prayer time fetching mechanism accurately retrieved times based on different dates and handled errors gracefully when data was unavailable.

- **Integration testing:** Integration testing played a crucial role in ensuring that different parts of the application worked together seamlessly. Using Postman, API endpoints were extensively tested to validate the interaction between the frontend and backend. This included sending erroneous inputs to ensure robust error handling and verifying that responses were correctly formatted and contained all necessary data. For instance, the API responsible for event creation was tested to confirm that it not only stored data correctly but also rejected invalid event details according to predefined rules.
- **Usability testing:** Usability testing was conducted using the Think-Aloud Protocol, where users including ISOC committee members interacted with the frontend. They performed predefined tasks such as creating an event or navigating to a specific Quran page. This method helped identify user interface issues and gather qualitative feedback on the app's usability. The insights gained were invaluable for refining the user interface, enhancing the overall user experience by addressing the practical and aesthetic concerns raised during these sessions.

The comprehensive testing approach adopted for the Northumbria ISOC app encompassing unit, integration, usability, and user acceptance testing has not only bolstered the application's reliability and user satisfaction but also underscored its readiness for deployment. Each testing phase was carefully chosen based on the specific validation needs at different stages of development, ensuring a robust and user-centric final product. The testing processes have successfully mitigated many potential faults, though as with any complex software, minor issues may still arise, necessitating ongoing monitoring and future updates (Refer to appendix C for full table reference of user metrics and feedback).

Prayer Times

Unit testing was critical for the 'fetchPrayerTimes' function within the prayer times feature (See Figure 41 & 42), as this function is responsible for retrieving daily prayer times from the database and formatting them correctly for frontend use. This function's accuracy is pivotal, not only for displaying all prayer times but also for identifying the next specific prayer time based on the current date and time.

```php
public function testFetchPrayerTimesForToday()
{
    // Mock the expected data
    $expectedData = [
        'Fajr' => '03:14:00',
        'FajrIqamah' => '03:19:00',
        'Dhuhr' => '13:03:00',
        'Sunrise' => '05:09:00',
        'DhuhrIqamah' => '13:08:00',
        'Asr' => '17:17:00',
        'AsrIqamah' => '17:22:00',
        'Maghrib' => '20:59:00',
        'MaghribIqamah' => '21:04:00',
        'Isha' => '22:53:00',
        'IshaIqamah' => '22:58:00',
    ];

    // Create an instance of the controller
    $controller = new PrayerTimesController();

    // Call the method to test
    $result = $controller->fetchPrayerTimesForToday(date('d/m/Y'));

    // Assert that the method returns the expected result
    $this->assertEquals($expectedData, $result);
}
```

Figure 41. unit test for the 'fetchprayertimes' function which passed the test.

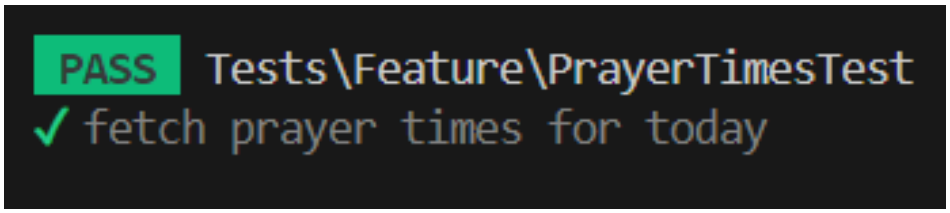


Figure 42. Confirmation of the fetchPrayerTimes function's reliability through successful unit test executions.

To validate this functionality, unit tests were designed with predefined data arrays that represented expected outcomes. These tests successfully confirmed that the function consistently retrieved and formatted prayer times accurately. This reassurance was bolstered by integration testing using Postman, which simulated API calls to assess the system's response under operational conditions. These tests were not only aimed at confirming data accuracy but also at evaluating performance metrics such as response times and data integrity across requests.

Further integration testing ensured seamless interaction between the backend and the frontend. By employing the API's GET method, the tests verified that both the complete daily prayer schedules and the next prayer times were fetched correctly by the frontend, confirming the API's reliability and efficiency (See Figure 43). These tests were conducted regularly throughout development to guarantee consistent functionality at various times and dates.

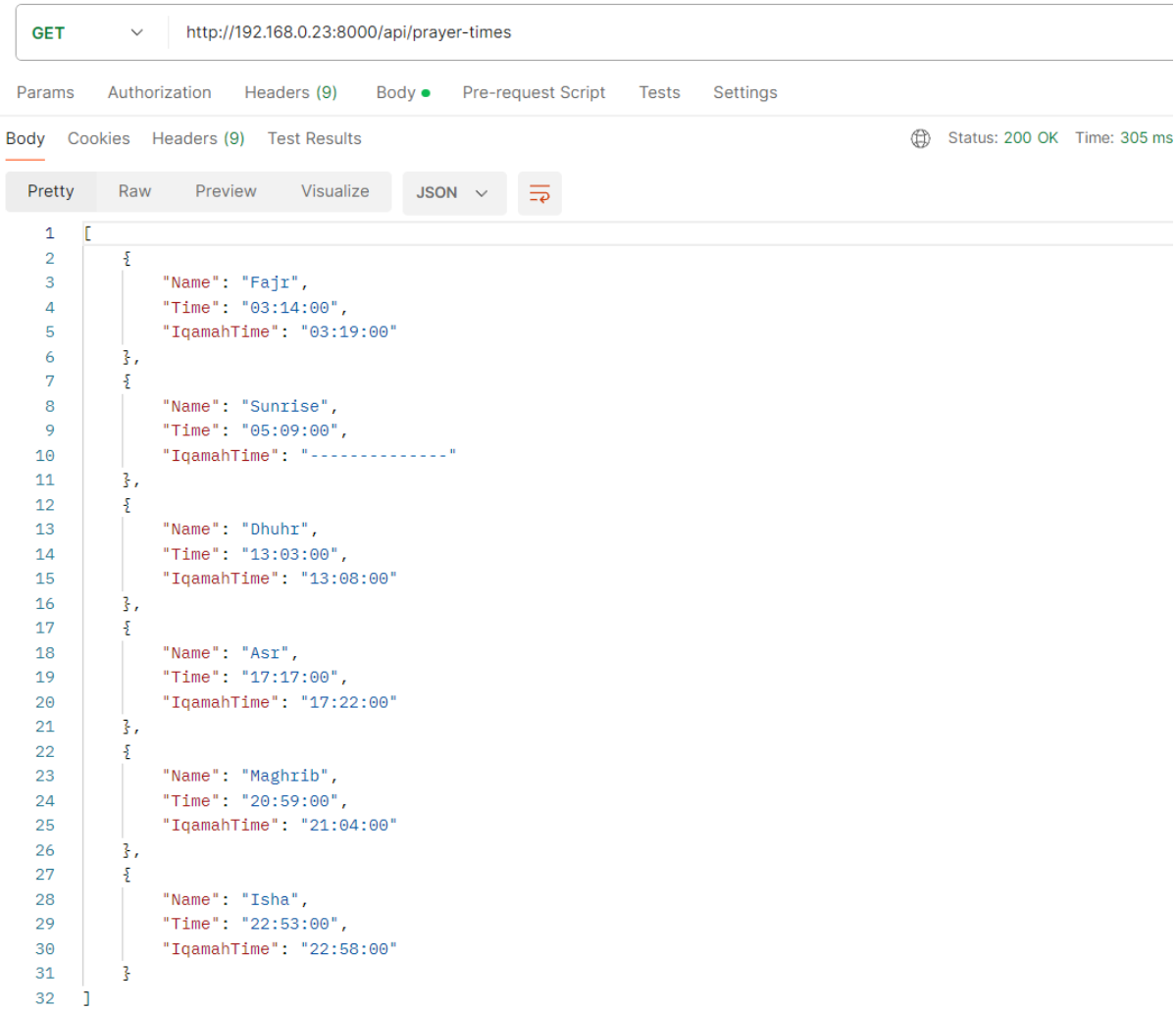


Figure 43. API response times and data accuracy tested using Postman, under various network conditions.

Usability testing provided invaluable insights directly from end-users, primarily focusing on the application's navigational intuitiveness and design. During these sessions, a small, diverse group of five randomly selected users from Northumbria ISOC's Muslim prayer facility engaged in specific tasks that tested the app's user interface and functionality. These users were tasked with identifying the next prayer time and its name, as well as determining the last prayer of the day and its corresponding Iqamah time. Through the think-aloud protocol, one user remarked, "I can see the next prayer is Maghrib at 6:30 PM; it's clearly visible," while another noted, "The Iqamah for Isha is set 5 minutes after the prayer time. It's helpful to have both times displayed together, so we can pray in congregation."

This direct feedback highlighted areas for improvement, such as the countdown timer's font size, which users found too small. This issue was promptly addressed by increasing the font size to enhance readability, as shown in Figure 44. Moreover, based on user suggestions, a Qibla direction feature was integrated using Google's Qibla Finder, significantly enriching the app's utility by directly addressing user needs and enhancing the practicality of the prayer times feature (See Figure 44).

These enhancements, driven by user feedback, led to substantial design improvements, making the app more engaging and user-friendly. Usability testing was instrumental in revealing these practical

enhancements, confirming the functional adequacy of the feature and fostering a user-centric development approach.

Figure 44. Enhancements post-usability testing: Updated user interface showing increased font size for countdown and new Qibla direction feature.

Quran

Unit testing was instrumental in verifying the Quran feature's core functionalities, particularly the retrieval of lists and page queries. These tests adopted a methodology like that used for the prayer times feature, employing predefined data arrays that mirrored the database contents. This approach validated whether the system could accurately fetch the requested Juz, Surah list, and page list, consistently returning the expected results (See Figure 45 & 46). During one of these tests, an unexpected failure revealed a critical vulnerability in the error handling mechanism when querying non-existent pages, leading to code enhancements to robustly manage such exceptions from the frontend.

```php
public function testListJuz()
{
    // Mock the expected Juz data
    $expectedJuz = [
        [
            'juz_number' => 1,
            'juz_name' => '(الحمد لله)',
        ],
        [
            'juz_number' => 2,
            'juz_name' => '(سيقول السفهاء)',
        ],
    ];

    // Create an instance of the controller
    $controller = new QuranController();

    // Call the method to test
    $result = $controller->listJuz();

    // Convert the actual result to an array
    $actualJuz = $result->original->toArray();

    // Assert that the method returns the expected result
    $this->assertEquals($expectedJuz, array_slice($actualJuz, 0, count($expectedJuz)));
}
```

Figure 45. Unit test for retrieving specifically the Juz list.

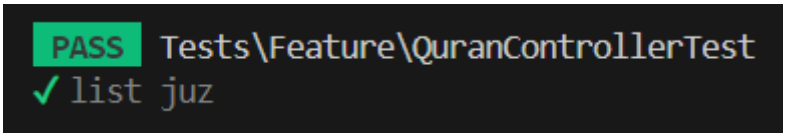


Figure 46. successful retrieval of the Juz List

Following the unit tests, integration testing was conducted using Postman, rigorously examining all queries across varying database states from individual Ayahs to complete pages. This iterative testing and development process enhanced the overall system's robustness, ensuring API endpoints reliably returned the correct data under diverse conditions. Successful integration tests were confirmed by satisfactory HTTP response statuses and the accurate retrieval of data, as evidenced in Postman logs (See Figure 47).

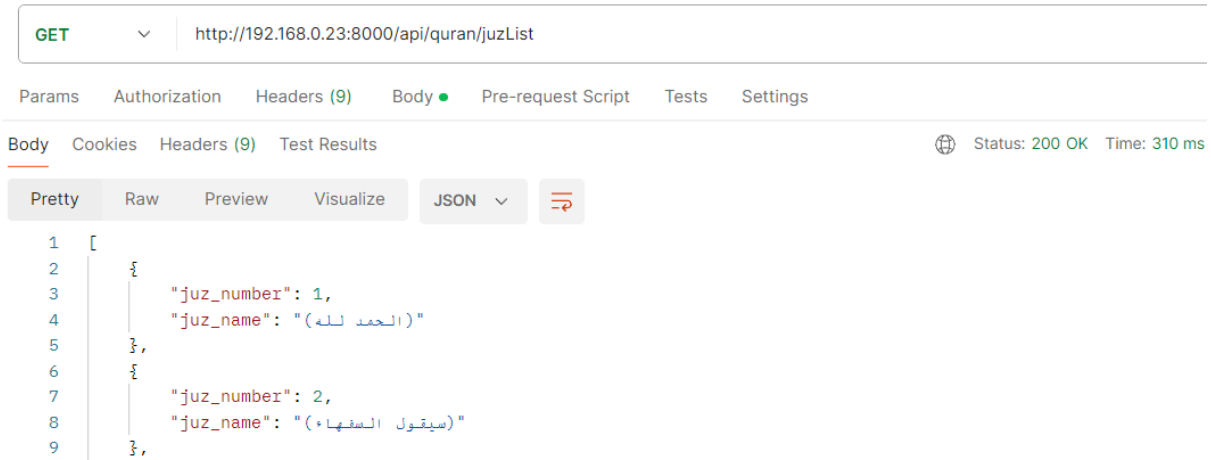


Figure 47. API endpoint retrieving the correct data in JSON format with a 200 response status and a quick response time of 310ms.

Usability testing then took centre stage, involving 14 randomly selected participants from the Muslim prayer facility at Northumbria University. This phase focused on authenticating the Quran text and evaluating the navigational experience across different tabs. Participants were tasked with specific actions, such as accessing Surahs or Juz, to assess the intuitive design and functionality of the interface. During these sessions, one participant suggested, "It would be really helpful to have a search box, so we can directly jump to a Surah or Page without scrolling," highlighting a user-driven improvement to enhance navigational efficiency. This feedback was pivotal, prompting the addition of a search feature that significantly augmented user satisfaction and interaction with the Quran feature (See Figure 48).

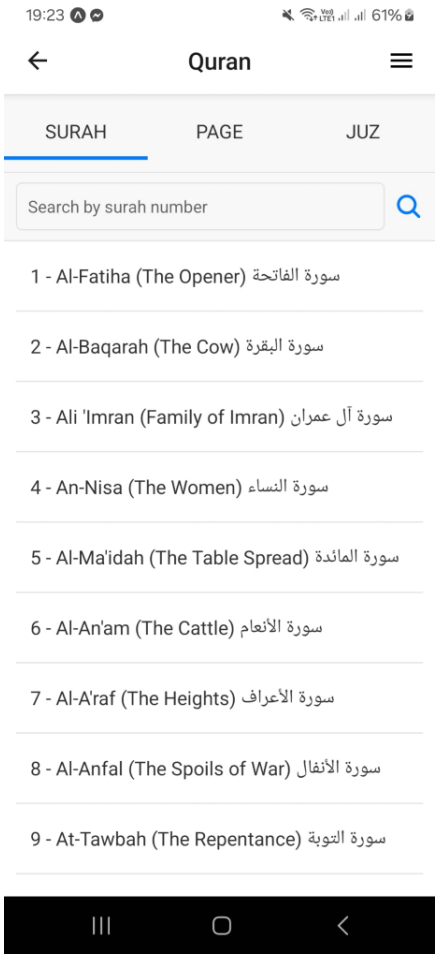


Figure 48. Quran feature with an intuitive 3 tab layout along with the requested search box for quick navigation.

This comprehensive testing strategy not only fortified the technical foundation of the Quran feature but also refined its usability aspects, making the application more aligned with the needs and preferences of its user base. Through these meticulous testing phases, the Quran feature has been polished to meet and exceed the expectations of the Northumbria University Muslim community, ensuring a reliable and enriching user experience.

Events

For the Events feature, integration testing focused extensively on administrative functions to ensure that only authorised users specifically administrators could access and perform create, edit, and delete operations. This verification was handled at the frontend, where administrative controls including the "Add," "Edit," and "Delete" buttons were conditionally displayed only to users with admin privileges. This design approach efficiently prevented unauthorised access at the user interface level, simplifying the backend validation processes.

Postman, a comprehensive tool for API testing, was instrumental in simulating various admin-level interactions to ensure that once access was granted, the backend correctly processed CRUD operations. DB Browser was utilised to directly observe the immediate database changes corresponding to these operations, providing a clear visual confirmation of the backend's accurate response to API calls (See Figures 49-52).

Integration testing encompassed:

- **Creating Events**: Confirming that event creation was facilitated smoothly for admins, with all details correctly captured in the database.
- **Editing Events**: Ensuring that modifications made by admins were instantly reflected in the database, maintaining data integrity and consistency.
- **Deleting Events**: Verifying the complete removal of events from the database to ensure clean deletions without residual data.

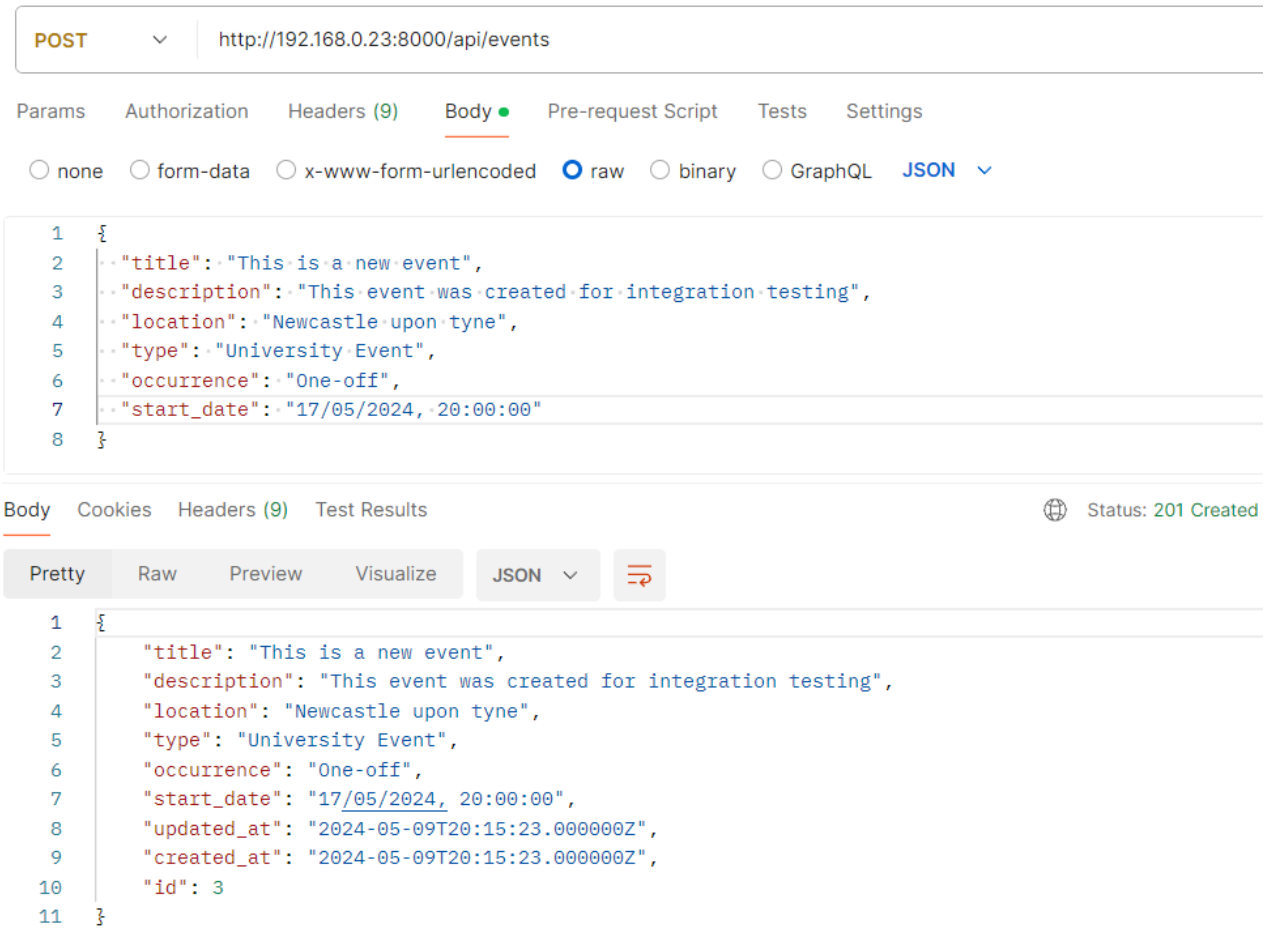
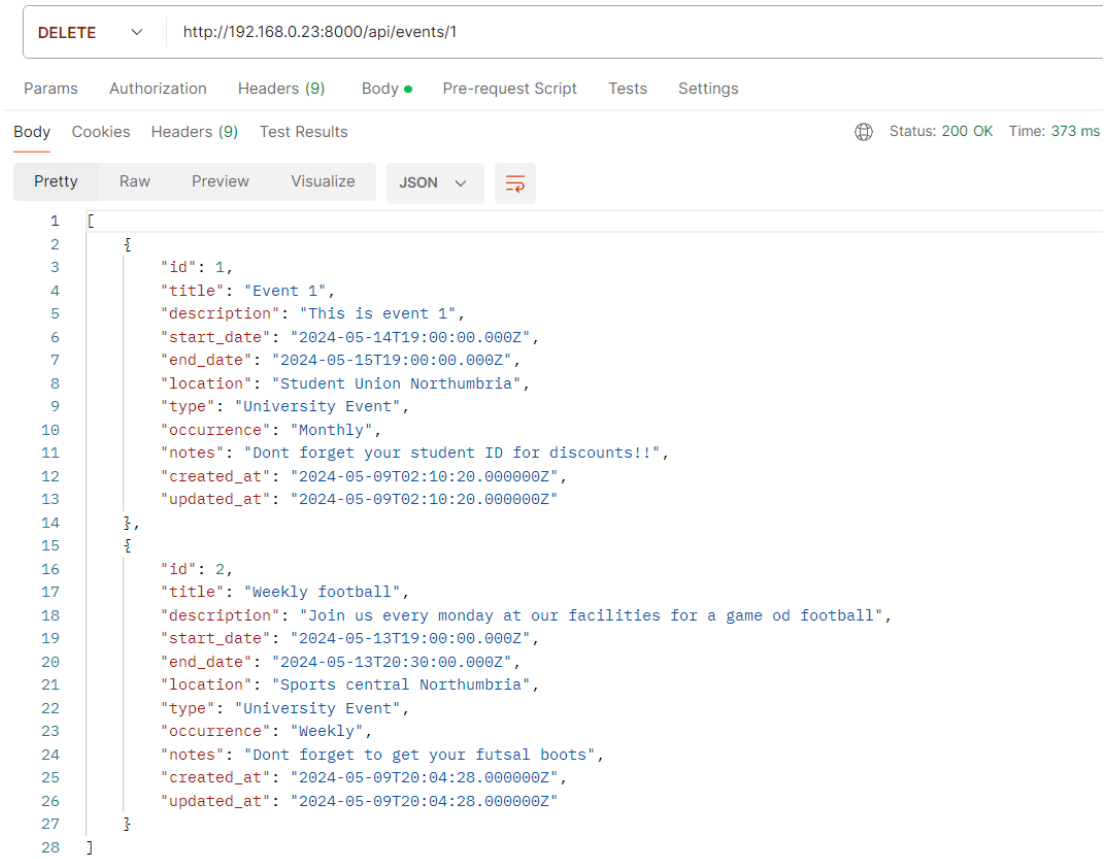


Figure 49. creating an event using the POST method successfully created indicated by the 200 response code shown by Postman.

```
    {
        "id": 2,
        "title": "Weekly football",
        "description": "Join us every monday at our facilities for a game od football",
        "start_date": "2024-05-13T19:00:00.000Z",
        "end_date": "2024-05-13T20:30:00.000Z",
        "location": "Sports central Northumbria",
        "type": "University Event",
        "occurrence": "Weekly",
        "notes": "Dont forget to get your futsal boots",
        "created_at": "2024-05-09T20:04:28.000000Z",
        "updated_at": "2024-05-09T20:04:28.000000Z"
    },
    {
        "id": 3,
        "title": "This is a new event",
        "description": "This event was created for integration testing",
        "start_date": "17/05/2024, 20:00:00",
        "end_date": null,
        "location": "Newcastle upon tyne",
        "type": "University Event",
        "occurrence": "One-off",
        "notes": null,
        "created_at": "2024-05-09T20:15:23.000000Z",
        "updated_at": "2024-05-09T20:15:23.000000Z"
    }
]
```


Figure 50. The new event clearly shown when retrieving the events using the GET method from the backend.




```
[
    {
        "id": 1,
        "title": "Event 1",
        "description": "This is event 1",
        "start_date": "2024-05-14T19:00:00.000Z",
        "end_date": "2024-05-15T19:00:00.000Z",
        "location": "Student Union Northumbria",
        "type": "University Event",
        "occurrence": "Monthly",
        "notes": "Dont forget your student ID for discounts!!",
        "created_at": "2024-05-09T02:10:20.000000Z",
        "updated_at": "2024-05-09T02:10:20.000000Z"
    },
    {
        "id": 2,
        "title": "Weekly football",
        "description": "Join us every monday at our facilities for a game od football",
        "start_date": "2024-05-13T19:00:00.000Z",
        "end_date": "2024-05-13T20:30:00.000Z",
        "location": "Sports central Northumbria",
        "type": "University Event",
        "occurrence": "Weekly",
        "notes": "Dont forget to get your futsal boots",
        "created_at": "2024-05-09T20:04:28.000000Z",
        "updated_at": "2024-05-09T20:04:28.000000Z"
    }
]
```


Figure 51. event id 1 is chosen to be deleted using the DELETE method.

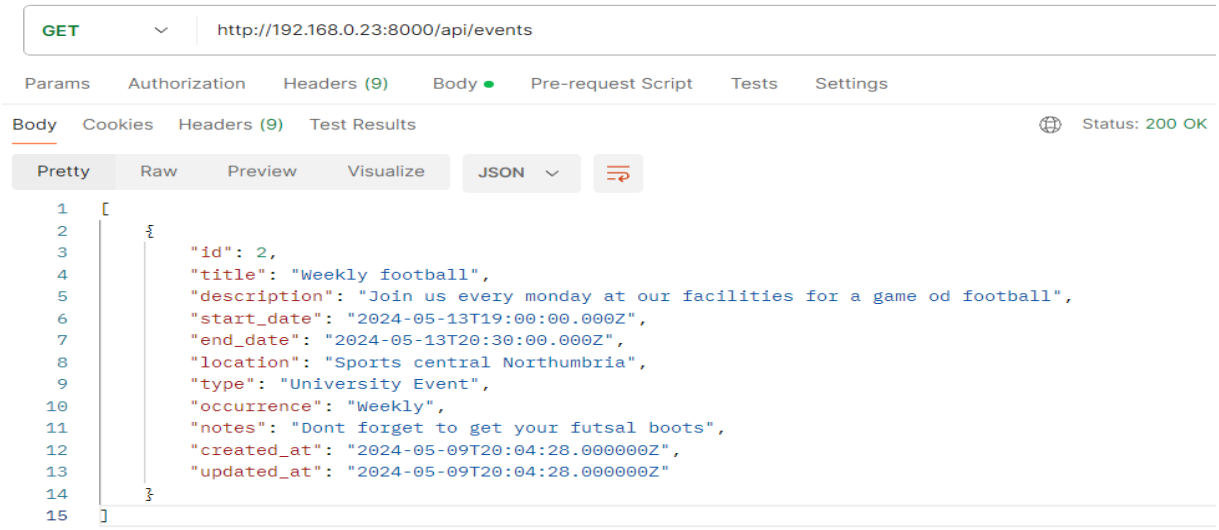


Figure 52. event id 1 is no longer available after successful deletion as shown only event id 2 is shown.

Simultaneously, usability testing was conducted with a broader user base, including ISOC committee members and a diverse mix of students and faculty, to assess the front-end functionality. Users were engaged in tasks that mirrored real-world usage scenarios:

- **Viewing Events**: Participants navigated the event list to find specific details, such as the timing and location of weekly football games.
- **Creating and Managing Events**: Admin users tested the interface by setting up and modifying event details, evaluating the ease of input and control.

The feedback from usability testing was overwhelmingly positive, affirming that the event management system was not only functional but also user-friendly. Users particularly appreciated the mobile app-like interface, which provided a familiar and intuitive user experience (See Figure 53). This feedback was crucial in refining the feature, ensuring that it not only met the technical specifications but also aligned with user expectations and needs.

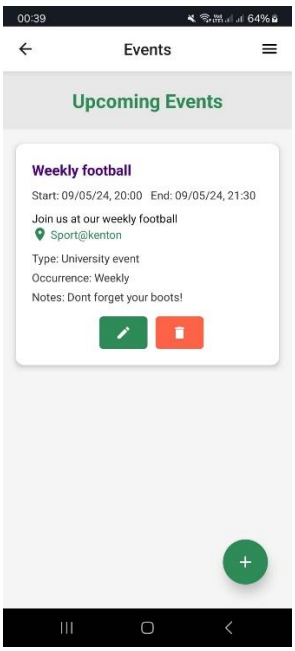


Figure 53. intuitive interface for admins to add, edit and delete events after affirmation from users during usability testing.

5. Evaluation

This evaluation chapter reviews the development and performance of the Northumbria ISOC app, drawing upon the Terms of Reference (ToR) as a benchmark to assess how well the final product met the defined user needs. The evaluation is structured into three main sections: firstly, assessing the product, secondly, examining the project process and management and thirdly, evaluating the impact and user engagement. Each section aims to detail the successes and challenges encountered during the project lifecycle.

The initial aim of the project, as outlined in the ToR, was to create a comprehensive digital solution for the Muslim community at Northumbria University. This solution was expected to provide accurate prayer times, details of ISOC committee events, and other relevant information to enhance the university experience for Muslim students and faculty. The evaluation criteria focus primarily on user satisfaction and the app's effectiveness in meeting these community-specific needs.

Throughout this section, we will explore how the project not only met these expectations but also expanded its feature set to include additional resources like the Quran, which was perhaps the biggest addition to the application, mosque locations, daily remembrance phrases (athkar), and an innovative support system named “Ask Me.” These enhancements aimed to enrich the user experience significantly, going beyond the basic functionalities initially projected. This introduction sets the stage for a detailed exploration of these elements and their impact on the target community.

End-Product

The Northumbria ISOC app robustly meets its core functionality requirements, providing reliable prayer times sourced from multiple local mosque timetables and a dynamic event management system. Admins can seamlessly add, edit, and delete events, ensuring regular users have up-to-date access, all supported by robust backend authentication. The pivotal shift to React Native enhanced cross-platform compatibility, crucial for accommodating the diverse device ecosystem at Northumbria University. This decision, driven by React Native’s widespread community support and rich libraries, was essential for maintaining a cohesive codebase, optimising both development time and cross-platform user experience.

User feedback highlighted the app’s intuitive design, likening it to familiar applications and underscoring its ease of use. The incorporation of common navigational elements and a calming colour scheme contributed to an inclusive and accessible user interface. Furthermore, user-driven enhancements such as the Qibla finder and athkar features with counters and progress bars were added, reflecting the app’s responsiveness to user needs and boosting overall engagement.

The introduction of additional features like mosque locations, daily remembrance phrases (athkar), and the "Ask Me" support system received substantial positive feedback, especially the inclusion of the Quran, which was particularly praised. These features, developed in response to user feedback and ISOC committee recommendations, affirm the app’s alignment with community needs and expectations.

User feedback has been instrumental in shaping the development of the Northumbria ISOC app. During testing phases, user inputs were actively solicited and significantly influenced subsequent iterations. Implementations like the athkar counter and progress bars for tracking completions were directly integrated in response to user suggestions, enhancing both the app’s functionality and interactivity. These adjustments not only improved the app's usability but also its alignment with the actual needs and practices of the community, as evidenced by increased user satisfaction and engagement metrics.

While the app has significantly improved functionality and user interaction, future enhancements like dynamic prayer time calculations and automated event notifications could further increase its utility and inclusiveness. These features, however, must be carefully designed to respect community diversity and user preferences. Moving forward, adopting a collaborative approach with local religious authorities and considering user feedback on notifications will be crucial. Additionally, integrating HCI research on adaptive user interfaces could enhance personalisation, catering to the evolving needs of the Northumbria University Muslim community. Such continuous improvements align with the project's ethos and pave the way for even greater user engagement and satisfaction.

Additionally, the event management functionality of the app could be expanded to include automated notifications. This would alert users about new events and remind them of upcoming activities, thereby increasing engagement and participation. Implementing such notifications would require careful consideration of user preferences to avoid excessive alerts, balancing informativeness with user convenience.

From a broader perspective, the app's development offers valuable insights into Human-Computer Interaction (HCI) principles, particularly in designing technology that is culturally sensitive and community-focused. The project underscores the importance of user-centred design and iterative feedback loops in creating software that not only meets functional requirements but also resonates with users on a personal level. Future iterations of the app could benefit from ongoing HCI research, particularly around adaptive user interfaces and personalised user experiences, which could further refine how the app meets diverse user needs within a dynamic university environment.

These reflections not only highlight areas for future development but also align with the project's continuous improvement ethos, suggesting a roadmap that could lead to even greater impact and user satisfaction.

Project Process and Management

Initially, a strict timeline was set in the ToR, but as development progressed, it became evident that adaptations were necessary. The engagement with the ISOC committee introduced additional features and required iterative feedback loops that, while extending the timeline, significantly enriched the app’s functionality. These changes, prompted by both committee insights and user feedback, led to a more user-centric application, highlighting the project's flexibility and responsiveness to emerging needs.

Human resources, particularly participants from the Muslim Prayer Facility, played a crucial role in testing and refining the app. Their insights were invaluable in enhancing inclusivity and functionality. However, implementing the 'Ask Me' chatbot feature presented funding challenges, as initial tests were self-financed. This necessitated considerations for scalable funding solutions, such as usage limits or ISOC committee sponsorship, illustrating the practical financial constraints often encountered in project management.

The most significant challenges included ensuring the authenticity and accuracy of religious content, managing diverse feature suggestions from committee members, and restricted access to the university’s database which limited potential app features. Each challenge required careful management, from rigorous source verification to consensus-building for feature inclusion, showcasing effective problem-solving and project agility. The proposed QR code feature for mosque access, although not implemented, represents a thoughtful consideration of user needs and practical utility.

Adopting an agile project management approach allowed for continuous refinement through prototype iterations and user feedback. This methodology proved essential in navigating the complexities of app

development and ensuring the final product was both intuitive and well-aligned with user expectations.

A key lesson learned was the importance of building in flexibility and a more detailed initial planning phase. The experiences highlighted the need for a thorough understanding of technical and feature feasibility early in the project. Insights from broader research, including the significance of the Quran in similar religious apps, reinforced the decision to include it as a central feature, demonstrating the value of external research in underpinning project decisions.

The project also provided insights into broader HCI principles, particularly the importance of user-centric design. Consistency and navigability emerged as crucial factors for app usability, echoing findings from external studies on app development.

Impact and User Engagement

Usability testing for the Prayer Times functionality shows a high level of satisfaction, with most users finding the feature easy to use and responsive. Notably, the integration of a countdown and highlighter was well-received, evidenced by a user rating of 5 and a quick task completion time (User ID 1). However, there were suggestions for improvement, such as increasing the font size to enhance readability (User ID 2), which indicates a need for slight adjustments to meet user preferences better. Feedback also led to the addition of a Qibla direction feature, showcasing the app's responsiveness to user needs. This iterative approach, as seen from the testing iterations, helped refine the feature to better align with user expectations and enhance usability.

The Quran functionality testing reveals insights into the navigational aspects and the effectiveness of implemented features like the search box and page tabs. While users appreciated the quick access to specific pages and Surahs, challenges were noted with scrolling to distant pages, suggesting a need for a more direct navigation method (User ID 3 and User ID 4). The introduction of a search box in later iterations significantly improved performance, underscoring the importance of iterative feedback in feature development. Users rated the navigational changes positively, affirming the feature's evolution in meeting user needs (User IDs 8, 9, 11).

Feedback on the Event Management tasks was overwhelmingly positive, particularly in how events are created, edited, and deleted. Initial iterations highlighted some issues, such as the non-persistence of data during edits (User ID 1, Edit Event), which were promptly addressed in subsequent updates, enhancing the user experience and streamlining interactions. The final iteration saw users enjoying the intuitive controls and the seamless integration of functional enhancements, like direct links to Google Maps for event locations, which significantly enhanced user satisfaction (User ID 3, Where is the Location).

The testing data not only validates the app's current functionality but also highlights areas for improvement. For instance, the need for more accessible font sizes and direct page access in the Quran feature could be addressed in future updates to enhance user experience. Additionally, the iterative feedback has proven essential in refining features to better suit user needs, as evidenced by the enhancements from user suggestions.

This data-driven approach provides a robust framework for ongoing development. By continuing to leverage user feedback and integrate iterative improvements, the app can maintain high user engagement and satisfaction. Future developments could include more personalised features, such as customisable prayer notifications and enhanced accessibility settings, to cater to a wider user base.

Overall, the user testing phase has been instrumental in shaping the app's development, with significant improvements made in response to user feedback. The insights gained from this phase should be used to inform continuous development efforts, ensuring the app remains relevant and

highly functional for its users (Refer to appendix C for full table reference for user metrics and feedback).

6. Conclusions and Recommendations

The development of the NUMosque app set out to significantly enhance the religious and communal experiences of Muslim students at Northumbria University. The project aimed to create a comprehensive digital solution tailored to the needs of this community, addressing specific functionalities such as prayer times, Quran access, and event management. Throughout its development phase, the app has shown considerable promise in meeting these needs, as evidenced by extensive user testing and feedback.

The prayer times feature was particularly well-received, offering accuracy and ease of use that are expected to facilitate daily observances for users. The Quran feature, with its enhanced search capabilities and user-friendly navigation, has the potential to become an indispensable tool for users seeking easy access to religious texts. Similarly, the event management functionality evolved from a basic listing feature to an interactive platform that could significantly increase user engagement with community events.

The project has achieved several key milestones that align closely with the initial objectives. Designed to provide practical, everyday utility, the app incorporates features that were refined and expanded based on user feedback, indicating a broader and more nuanced understanding of community needs than initially anticipated.

The adoption of agile development practices allowed the project team to adapt swiftly to emerging requirements and incorporate real-time feedback into the development process. The selection of React Native enabled efficient cross-platform development, demonstrating the effectiveness of these tools in building flexible and responsive applications.

To build on the current groundwork and maximise the app’s impact upon potential future deployment, several enhancements are recommended:

- Dynamic Prayer Time Algorithm: Consider collaborating with local religious authorities to develop a widely accepted method for calculating prayer times, enhancing the app's inclusivity and accuracy.
- Advanced Notification Systems: Develop sophisticated notification systems for events to boost user engagement and ensure community members are well-informed about upcoming activities.
- Expansion of "Ask Me" Features: Investigate sustainable funding options for the "Ask Me" chatbot to expand its functionality and reach, providing broader support to users.

This project serves as a model for developing community-specific applications, demonstrating how targeted digital solutions can significantly enhance community engagement and participation. The insights gained from the iterative development and community-centred feedback approach should inspire and inform future projects aimed at addressing niche community needs.

In conclusion, while the NUMosque app has not yet been deployed, the extensive development and testing phases have shown that it is well-prepared to make a meaningful impact on the Northumbria Muslim community. The project's adherence to user-centred design principles and agile methodologies provides a robust foundation for future digital initiatives, potentially extending its benefits beyond the immediate university setting.

7. References

Alexander, B., Ashford-Rowe, K., Barajas-Murphy, N., Dobbin, G., Knott, J., McCormack, M., Pomerantz, J., Seilhamer, R., & Weber, N. (2019). EDUCAUSE Horizon Report: 2019 Higher Education Edition. https://library.educause.edu/-/media/files/library/2019/4/2019horizonreport.pdf?la=en&hash=C8E8D444AF372E705FA1BF9D4FF0DD4CC6F0FDD1

Oliveira, D.M.D., Pedro, L. and Santos, C. (2021). The use of mobile applications in higher education classes: a comparative pilot study of the students' perceptions and real usage. Smart Learning Environments, 8(1). doi:https://doi.org/10.1186/s40561-021-00159-6.

Lumor, Truth & Pulkkinen, Mirja & Hirvonen, Ari. (2021). The Actual Adoption and Use of Mobile Apps The Actual Adoption and Use of Mobile Apps: The Case of a Higher Education Context.

Pechenkina, E. (2017). Developing a typology of mobile apps in higher education: A national case-study. Australasian Journal of Educational Technology, 33(4). doi:https://doi.org/10.14742/ajet.3228.

Hameed, A., Ahmed, H.A. and Bawany, N.Z. (2019). Survey, Analysis and Issues of Islamic Android Apps. Elkawnie, 5(1), p.1. doi:https://doi.org/10.22373/ekw.v5i1.4541.

Campbell, H. (2010). When Religion Meets New Media (1st ed.).

Asadullah, A., Yerima, B., & Yusuf, A. O. (2014). The Ethics of Information and Communication Technology : An Islamic Overview. International Journal of Information and Communication Technology Research, 4(2), 45–49.

Khan, E. A., & Shambour, M. K. Y. (2017). An analytical study of mobile applications for Hajj and Umrah services. Applied Computing and Informatics, 14(1), 37–47. https://doi.org/10.1016/j.aci.2017.05.004

Mohamed, N., Mantoro, T., Media, A., & Mahmud, M. (2016). Critical Socio-Technical Issues Surrounding Mobile Computing. In M. M. Teddy Mantoro, Media Ayu (Ed.), IGI Global. https://doi.org/10.4018/978-1-4666-9438-5

Foltz, F. and Foltz, F. (2003). Religion on the Internet: Community and Virtual Existence. Bulletin of Science, Technology & Society, 23(4), pp.321–330. doi:https://doi.org/10.1177/0270467603256085.

Campbell, H.A. (Ed.). (2012). Digital Religion: Understanding Religious Practice in New Media Worlds (1st ed.). Routledge. https://doi.org/10.4324/9780203084861

Campbell, H.A., Rule, F. (2020). The Practice of Digital Religion. In: Friese, H., Nolden, M., Rebane, G., Schreiter, M. (eds) Handbuch Soziale Praktiken und Digitale Alltagswelten. Springer VS, Wiesbaden. https://doi.org/10.1007/978-3-658-08357-1_38

Albert, B. and Tullis, T. (2013). Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. [online] Google Books. Newnes. Available at: https://books.google.co.uk/books?hl=en&lr=&id=bPhLeMBLEkAC&oi=fnd&pg=PP1&dq=Measuring+the+User+Experience:+Collecting

Lowdermilk, T. (2013). User-Centered Design: A Developer's Guide to Building User-Friendly Applications. [online] Google Books. 'O'Reilly Media, Inc.' Available at: https://books.google.co.uk/books?hl=en&lr=&id=XiX5bNJjW0kC&oi=fnd&pg=PR2&dq=ser-Centered+Design:+A+Developer%E2%80%99s+Guide+to+Building+User-

Friendly+Applications&ots=F81G8PeCRn&sig=IrpzxADtOyOwRdZ5pZDvma6SG7w&redir_esc=y#v=onepage&q&f=false

Richardson, J., Ormerod, T.C. and Shepherd, A. (1998). The role of task analysis in capturing requirements for interface design. Interacting with Computers, 9(4), pp.367–384. doi:https://doi.org/10.1016/s0953-5438(97)00036-2

Albert, B. and Tullis, T. (2013). Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. [online] Google Books. Newnes. Available at: https://books.google.co.uk/books?hl=en&lr=&id=bPhLeMBLEkAC&oi=fnd&pg=PP1&dq=Measuring+the+User+Experience:+Collecting

8. Appendices

Appendix A: Project Timeline

Appendix B: Requirements

Functional Requirements

Requirement ID	Name	Description	Priority	Source	Acceptance Criteria
FR1	User Authentication	Users can create an account and log in to access the app.	Essential	Industry Standards	Users can register and log in with no errors, account details are stored and retrieved securely.
FR2	Prayer Times Display	Display daily prayer times and iqamah times on the homepage.	Essential	Literature Review	Prayer times match with the local mosque's schedule, automatic updates for day-to-day prayer times.
FR3	Quran Access	Users can read the Quran text and navigate through surahs.	Essential	ISOC Committee feedback, Literature Review	Text is legible and searchable by surah, juz, and page; user can navigate easily between sections.
FR4	Event Calendar	Users can view upcoming events organised by the ISOC.	Essential	User feedback, ISOC Committee feedback	Events are listed in chronological order with all essential details
FR5	Real-Time Inquiries	Users can ask questions through the 'Ask Me' feature.	High	Developer Initiative	Queries receive responses within a certain timeframe, user satisfaction with answers is high.

FR6	Location Services	Mosque locations are provided with directions via Google Maps.	High	ISOC Committee feedback	Directions are accurate, and links open directly in Google Maps.
FR7	Athkar Reminders	App provides a section for daily reminders for Islamic supplications.	Low	Developer Initiative, Competitor Analysis	The Athkar section is clearly labelled and accessible

Table 1. Functional Requirements

Non-functional Requirements

Requirement ID	Name	Description	Priority	Source	Acceptance Criteria
NFR1	Security	The app must implement a secure user authentication and authorisation mechanism to protect user data and privacy.	High	Project Specification	The app correctly utilises JWT tokens to grant access solely to authorised users, with robust validation.
NFR2	Reliability	The app must perform consistently under specified conditions.	High	Industry Standards	99.9% uptime, full data recovery within 1 minute
NFR3	Performance	The app must load and respond quickly to user input.	High	Project Specification	Response time under 2 seconds
NFR4	Maintainability	The app should be easy to maintain and update.	Medium	Developer Guidelines	Modular architecture and documented code
NFR5	Scalability	The app must be able to handle an increasing number of users.	Medium	Growth Projections	Efficient performance under scaling user base, with provisions for future expansion.
NFR6	Usability	The app must be user-friendly and accessible to all users.	High	Accessibility Standards	positive user test feedback.
NFR7	Compatibility	The app must be compatible across different devices and operating systems.	High	Project Specification	Smooth operation on latest and last two versions of Android and iOS.

Table 2. Non-functional Requirements

Field	Description
Requirement ID	FR1
Use Case Name	User Authentication
Objective	To securely verify the identity of a user attempting to access the Northumbria ISOC app features, utilising Firebase authentication services.
Actors	App User (Student/Faculty), Firebase Authentication Service
Preconditions	1. The user must have the Northumbria ISOC app installed on their device.

	2. Firebase Authentication service must be correctly configured and operational.
Postconditions	1. Successful authentication grants the user access to their personalised app environment. 2. Failed authentication keeps the user at the login screen with an error message displayed.
Trigger	User selects 'Log in' or opts to 'Register' through the app's initial screen.
Normal Flow	1. The user opens the app and is presented with the login screen. 2. The user inputs their email and password. 3. The app communicates with Firebase to validate the credentials. 4. Firebase Authentication confirms the validity of the credentials. 5. Upon successful validation, the user is granted access and redirected to the homepage. 6. If validation fails, Firebase sends an error notification, and the user is prompted to try again.
Alternative Flows	• **New User Registration**: 1. The user chooses 'Register'. 2. The user completes the registration form with necessary details. 3. The user submits the form, which is processed by Firebase. 4. Firebase registers the user and sends an email verification link. 5. The user verifies their email and is redirected to the login page. • **Forgot Password:** 1. The user selects 'Forgot Password'. 2. The user provides their email address linked to the account. 3. Firebase initiates a password reset process and emails a reset link. 4. The user resets their password using the link and logs in with the new credentials.
Exception Flows	1. Unable to communicate with Firebase due to network issues. 2. Registration attempt with an already used email. 3. An expired or invalid password reset link.
Special Requirements	1. Use HTTPS for all data transmissions involving Firebase. 2. Implement rate limiting and monitoring to prevent brute force attacks.
Assumptions	1. Users have access to the internet and their email accounts for registration and password recovery. 2. Firebase services are configured to comply with all relevant privacy laws including GDPR.
Frequency of Use	Users are required to authenticate upon first accessing the app. Once logged in, they can choose to stay logged in, bypassing authentication on subsequent app launches unless they explicitly log out.
Miscellaneous	1. Ensure the Firebase configuration adheres to the latest security standards. 2. Consider implementing multi-factor authentication to enhance security.
Related NFR	NFR1, NFR5

Field	Description
Requirement ID	FR2
Use Case Name	Display Prayer Times

Objective	To provide accurate and timely prayer schedules for the Muslim student community at Northumbria University, utilising a local SQLite database updated in real-time.
Actors	App User (Student/Faculty)
Preconditions	1. The user must be successfully authenticated and logged into the Northumbria ISOC app. 2. The local SQLite database must be up-to-date with the latest prayer times.
Postconditions	1. The user views the current day's prayer times updated according to their local time zone. 2. The user can access historical and future prayer times if needed.
Trigger	User navigates to the 'Prayer Times' section within the app.
Normal Flow	1. The user logs into the app and navigates to the 'Prayer Times' section from the main menu. 2. The app retrieves the current date and time from the user's device. 3. The app queries the SQLite database to fetch prayer times corresponding to the current date. 4. The app displays the prayer times on the user's device.
Alternative Flows	• **Update Prayer Times:** 1. The user requests to update prayer times. 2. The app checks for an available network connection. 3. If connected, the app updates the SQLite database with the latest times from a trusted online source. 4. The user views updated prayer times.
Exception Flows	1. Network issues prevent updating the prayer times. 2. The local database fails to retrieve the prayer times due to a corruption error.
Special Requirements	Utilise caching to reduce the need for frequent database queries.
Assumptions	1. Users are interested in seeing prayer times for the upcoming prayers. 2. The app has access to reliable and accurate source data for prayer times.
Frequency of Use	High, users may check the prayer times multiple times per day, especially close to each prayer interval.
Miscellaneous	1. Consider implementing notifications for upcoming prayer times to enhance user engagement. 2. Ensure the user interface for displaying prayer times is clear, concise, and easy to navigate.
Related NFR	NFR1, NFR3, NFR7

Field	Description
Requirement ID	FR3
Use Case Name	Quran Text Access
Objective	To provide users with an interactive and easily navigable Quran interface, allowing them to read and explore the Quran text digitally.
Actors	App User (Student/Faculty)
Preconditions	1. The user must be successfully authenticated and logged into the Northumbria ISOC app. 2. The app must have access to a robust API or local database containing the full text of the Quran.
Postconditions	The user successfully accesses the Quran text and can interact with different features such as searching, bookmarking, and note-taking.
Trigger	User selects the 'Quran' option from the main menu of the app.

Normal Flow	1. The user logs into the app and selects the 'Quran' section from the main menu. 2. The app presents options to choose Surah (chapters), Juz (parts), or a search function to find specific pages. 3. The user selects a Surah or Juz or searches for specific verses. 4. The app displays the selected Quranic text. 5. The user reads the selected text
Alternative Flows	1. The user uses the search bar to look up specific pages, or Juz. 2. The app displays all relevant results within the Quranic text, allowing the user to navigate directly to the desired section or verse from the search results.
Exception Flows	1. The API or local database fails to load the Quranic text due to network or technical issues. 2. Incorrect or missing translations or tafsir due to data errors.
Special Requirements	1. Utilise secure and reliable data transmission methods when fetching Quranic text from external sources. 2. Ensure that the interface accommodates various reading preferences and accessibility standards.
Assumptions	1. Users have a basic understanding of how to navigate digital content. 2. The source of the Quranic text is reliable and regularly updated to ensure accuracy.
Frequency of Use	High, as users may access the Quranic content daily for personal study and reflection.
Miscellaneous	1. Regular updates to the Quranic content and app features based on user feedback and technological advancements. 2. Potential integration of additional religious resources and scholarly works for comprehensive study.
Related NFR	NFR1, NFR2, NFR7

Field	Description
Requirement ID	FR4
Use Case Name	Event Calendar
Objective	To enable users to view and participate in community events through the app, enhancing community interaction and participation.
Actors	App User (Student/Faculty), Administrator
Preconditions	1. Users are authenticated and have appropriate permissions (administrators for managing events, general users for viewing events). 2. The app must be connected to a backend system capable of storing and retrieving event data.
Postconditions	1. Users successfully view the list of upcoming events. 2. Administrators can create, modify, or delete event listings as needed.
Trigger	User selects the 'Events' option from the app menu.
Normal Flow	1. The user opens the app and navigates to the 'Events' section. 2. The app displays a calendar with scheduled events. 3. Users click on an event to view detailed information, including time, location, description, and participation instructions.
Alternative Flows	• **Adding New Events (Administrators only):** 1. Administrator selects the option to add a new event. 2. Fills in details such as event title, date, time, location, and description. 3. Submits the event, which is then added to the calendar. • **Editing Existing Events (Administrators only):**

	1. Administrator selects an existing event to edit. 2. Updates the necessary details and saves the changes. • **Deleting Events (Administrators only):** 1. Administrator selects an event to delete. 2. Confirms the deletion, and the event is removed from the calendar.
Exception Flows	1. Errors in data submission such as incomplete event details. 2. Failure to load events due to network issues. 3. Unauthorised attempts to modify or delete events.
Special Requirements	1. Ensure secure authentication for administrative actions. 2. Implement user-friendly date and time pickers for ease of event entry. 3. Optimise calendar display for different devices and orientations.
Assumptions	1. Users have varying degrees of technical skill; therefore, the interface should be intuitive and simple to navigate. 2. Administrators are responsible for the accuracy and timeliness of the event information provided.
Frequency of Use	Frequently used as community events are regularly organised and attended by users.
Miscellaneous	1. Consider implementing notifications for upcoming events to increase engagement. 2. Regularly update the system to handle an increasing number of events and users.
Related NFR	NFR1, NFR2, NFR3

Field	Description
Requirement ID	FR5
Use Case Name	Ask Me Feature
Objective	To provide instant responses to user queries through the 'Ask Me' feature, utilising the OpenAI API for generating intelligent and contextually relevant answers.
Actors	App User (Student/Faculty), OpenAI API
Preconditions	1. The user must be logged into the Northumbria ISOC app. 2. The OpenAI API must be integrated and operational within the app.
Postconditions	1. User receives a prompt and relevant response to their query. 2. The system logs the inquiry and response for potential future analysis.
Trigger	User submits a query through the 'Ask Me' feature.
Normal Flow	1. User navigates to the 'Ask Me' section of the app. 2. User types a question into the inquiry field and submits it. 3. The app sends the query to the OpenAI API. 4. OpenAI processes the question and generates a response based on learned models and data. 5. The response is sent back to the app and displayed to the user.
Alternative Flows	1. If the user is not satisfied with the response, they can ask a follow-up question or rephrase their query for clarity. 2. Steps 3-5 of the Normal Flow repeat until the user is satisfied with the answer.
Exception Flows	1. The OpenAI API is temporarily unavailable due to network issues or maintenance. 2. The API fails to generate a response due to an unprocessable query.
Special Requirements	1. Implement proper authentication and authorisation checks to ensure that queries are sent by authenticated users. 2. Ensure all data exchanged with the OpenAI API is encrypted to maintain confidentiality and integrity.

Assumptions	1. Users have basic knowledge of how to interact with digital inquiry systems. 2. The OpenAI API remains up-to-date with the latest data security standards.
Frequency of Use	Potentially high, as users may utilise the feature regularly to get information related to Islamic practices and community events.
Miscellaneous	Monitoring and logging mechanism should be in place to analyse the usage patterns and improve the system based on user feedback.
Related NFR	NFR1, NFR3

Field	Description
Requirement ID	FR6
Use Case Name	Location Service
Objective	To provide users with accurate locations and directions to mosques via integration with Google Maps, facilitating easy access to prayer facilities.
Actors	App User (Student/Faculty)
Preconditions	1. The user must be authenticated and have access to the location services within the app. 2. Google Maps integration must be active and operational within the app.
Postconditions	Users are able to view mosque locations and receive directions to these locations effectively within the app.
Trigger	User selects the mosque location option on the app.
Normal Flow	1. The user accesses the location services feature within the app. 2. The app displays a list of nearby mosques or allows the user to search for a specific mosque. 3. The user selects a mosque from the list. 4. The user clicks on the directions button, which opens Google Maps and provides turn-by-turn navigation to the selected mosque.
Alternative Flows	1. In addition to viewing directions through Google Maps, the user has the option to copy the mosque address. 2. User can paste the copied address into any other map or navigation application of their choice for directions.
Exception Flows	1. Google Maps API fails to load or returns an error. 2. Incorrect location data leads to inaccurate directions. 3. User's device GPS is disabled, preventing location detection.
Special Requirements	1. Ensure that location data used by the app is regularly updated and accurate. 2. Interface with Google Maps must maintain compliance with Google's API usage standards.
Assumptions	1. Users have functional GPS and internet access on their devices to use Google Maps. 2. Google Maps remains a reliable provider of mapping and navigation services.
Frequency of Use	Frequent use, especially during Islamic prayer times and on Fridays.
Miscellaneous	Provide user feedback mechanism to report any inaccuracies in mosque locations or directions.
Related NFR	NFR3, NFR5

Field	Description
Requirement ID	FR7
Use Case Name	Athkar Reminders

Objective	To provide users with daily Islamic supplications through timely reminders within the app, enhancing their spiritual practice.
Actors	App User (Student/Faculty)
Preconditions	1. The user has enabled notifications for the Athkar Reminders within the app settings. 2. The app has sufficient data to schedule reminders based on user preferences or default settings.
Postconditions	1. Users receive timely notifications for Athkar at specified times throughout the day. 2. Users can view and engage with daily supplications directly from the notification or within the app.
Trigger	A scheduled time for an Athkar reminder reaches.
Normal Flow	1. The Athkar Reminders feature fetches the day's supplications from the local database. 2. The app schedules notifications based on the user's set preferences or default times. 3. At the scheduled times, the app sends a notification to the user's device. 4. The user taps on the notification to view the full supplication in the app. 5. The user can read the supplication
Alternative Flows	• **User Disables Notifications:** 1. If the user disables notifications, they need to manually open the app to view the daily Athkar.
Exception Flows	1. The device fails to receive notifications due to connectivity issues or system settings that block app notifications. 2. Errors in the database prevent the fetching of accurate Athkar data.
Special Requirements	1. Ensure all notifications are delivered in a respectful and culturally appropriate manner. 2. Use local storage to keep the Athkar data accessible even when offline.
Assumptions	1. Users are interested in receiving daily reminders for religious practices. 2. Users have provided the necessary permissions for notifications.
Frequency of Use	Daily reminders at predetermined times based on user settings or default configurations.
Miscellaneous	1. Periodically update the database of supplications to maintain relevance and accuracy. 2. Provide options for customisation in the type and timing of reminders to accommodate personal schedules and preferences.
Related NFR	NFR4, NFR6

Appendix C: Additional features design, implementation and testing

Design

Ask Me

In recognising the diverse needs of new and international students, particularly those navigating unfamiliar environments and cultures, the Northumbria ISOC app incorporates a vital support system in the form of a chatbot. Leveraging the advancements in artificial intelligence, this feature utilises the OpenAI API to provide users with personalised assistance and guidance tailored to their specific queries.

During user testing, significant interest was observed in this feature, highlighting its potential to serve as a valuable resource for Muslim students seeking support and guidance. Notably, research indicates a lack of specialised support for religious students, particularly in their transition to new environments or relocation from abroad to Newcastle.

The development process encountered certain challenges, notably in adapting the AI functionality to React Native, a framework distinct from the typical Python environment for which OpenAI API is designed. Despite these challenges, diligent efforts were made to ensure seamless integration and optimal performance within the app.

Additionally, the implementation of this feature necessitates consideration of the associated costs, particularly concerning the pricing structure of the OpenAI API. Future funding considerations by the committee may be required to sustain and further develop this support system or explore alternative solutions tailored to their specific needs and budgetary constraints.

In essence, the Ask Me feature represents a proactive approach to addressing the diverse support needs of Muslim students within the Northumbria ISOC community, demonstrating the app's commitment to enhancing the overall student experience and facilitating a smooth transition into university life.

Mosque Locations

The Mosque Locations feature serves as a valuable resource, especially for international students and faculty members, facilitating their access to nearby mosques in Newcastle. By providing precise directions and locations through integration with Google Maps, users can easily locate mosques in the vicinity for congregational prayers and other religious events.

While the inclusion of this feature stemmed from project ideation rather than formal research, user testing revealed its significant utility, particularly among international students. During testing, international students expressed appreciation for the ability to locate mosques where they could participate in congregational Friday prayers and observe daily prayers throughout the week.

In essence, the Mosque Locations feature enhances the Northumbria ISOC app's functionality by promoting socialisation and facilitating religious practices, thereby enriching the experience of international students and faculty within the Northumbria University community.

Athkar

The Athkar feature serves as a daily reminder for users to uphold their religious commitments, offering easy access to various Athkar based on the time of day. With clear and distinct sections labelled "Morning Athkar," "Evening Athkar," and "Before Sleep Athkar," users can swiftly navigate to specific sections through a simple and intuitive interface.

Implemented in both Arabic and English, the feature ensures accessibility to a wider user base and authenticity by sourcing content from a reputable religious website. However, the main challenge encountered during implementation was structuring and formatting the extensive data about different Athkar in the database for seamless retrieval to the frontend.

This feature's addition was prompted by research indicating the popularity of Athkar apps among Muslims, suggesting a genuine need to incorporate it into the Northumbria ISOC app. By integrating the Athkar feature, the app further fulfils its goal of providing a comprehensive digital solution for the Muslim community at Northumbria University, enhancing their religious practice and spiritual well-being.

Implementation

Ask Me Chatbot

This functionality didn't require extensive backend setup, as the API call and endpoint were seamlessly handled using the OpenAI API. This allowed for the integration of a chatbot feature, enabling users to pose queries and receive prompt responses courtesy of the GPT-3.5-turbo model. Serving as a foundational feature, it offers ample opportunities for future enhancements which could include:

- Saving Chat Histories:
    - Considering the addition of a feature to save chat histories would greatly enhance user convenience by allowing them to revisit and continue past interactions effortlessly. This could be implemented by integrating a database system to securely store conversation logs linked to user IDs, ensuring each interaction can be retrieved when needed. Key considerations would include implementing robust encryption to maintain privacy and adhering to data protection laws to ensure compliance and safeguard user information.
- Fine Tuning Responses:
    - Fine-tuning the chatbot to specialise in academic-related queries could transform it into a more valuable resource for students and faculty. This enhancement would involve retraining the existing GPT-3.5-turbo model on a curated dataset comprising academic texts and common university-related questions. The process would benefit from iterative feedback from academic staff, which would help refine the accuracy and relevance of the responses, ensuring the chatbot better addresses the specific needs of the academic community.

Inspired by prevailing trends in major software companies, this AI integration aligns with the objective of aiding users with their queries. In the context of the Northumbria ISOC app, it functions as a reliable support system, catering to inquiries relating to university life and religious commitments with relatively up-to-date information.

The implementation of this feature involves a simple yet effective process. When a user inputs a query, it triggers a request to the OpenAI API, passing along the user's question. This request is handled asynchronously, ensuring that the app remains responsive during the query processing (See Figure 54 for more details).

Upon receiving the user's query, the OpenAI API utilises the GPT-3.5-turbo model to generate a response. This model is trained on a vast corpus of text data, enabling it to understand and generate human-like responses across a wide range of topics.

Once the response is generated by the GPT model, it is returned to the app via the API. The app then displays the response to the user, completing the interaction loop.

```
const handleSend = async () => {
  const userQuestion = input;
  setInput('');

  setConversation([...conversation, { type: 'user', text: userQuestion }]);

  try {
    const response = await fetch('https://api.openai.com/v1/chat/completions', {
      method: 'POST',
      headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${apikey}`
      },
      body: JSON.stringify({
        model: "gpt-3.5-turbo",
        messages: [{
          role: "user",
          content: userQuestion
        }]
      }),
    });

    const json = await response.json();
    if (!response.ok) {
      throw new Error(json.error ? json.error.message : "Unknown error");
    }

    const botAnswer = json.choices[0].message.content.trim();
    setConversation(prev => [...prev, { type: 'bot', text: botAnswer }]);
  } catch (error) {
    setConversation(prev => [...prev, { type: 'bot', text: "Sorry, I couldn't understand that." }]);
  }
};
```

Figure 54. 'HandleSend' function which is where the user inputs a message which gets sent to the GPT model and returns with a response which is displayed to the user.

Implementing this feature entailed adapting the chatbot functionality from its conventional Python environment to the React Native development ecosystem. This transition necessitated a few adjustments to seamlessly incorporate the chatbot feature. The primary challenge revolved around ensuring compatibility and optimal performance within the React Native framework.

Moving to the frontend, a deliberate design approach was adopted to ensure user-friendly interactions. User input is distinguished by a blue colour scheme, while bot responses are highlighted in cream. This intuitive design choice echoes the layout conventions observed in popular communication platforms like Facebook and WhatsApp. It enhances user experience by clearly demarcating between user messages and bot-generated responses.

During the implementation phase, meticulous attention was devoted to testing the responsiveness and quality of the bot's interactions. This involved rigorous evaluation of response times, which are contingent on network speed and query complexity, as well as the accuracy and relevance of the responses generated. Through comprehensive testing, the functionality was fine-tuned to deliver a seamless user experience.

Additionally, the GPT-3.5-turbo model excels at generating human-like responses, which significantly enhances user engagement and satisfaction. By leveraging its advanced natural language processing capabilities, the chatbot can provide responses that are not only accurate but also well-articulated and contextually relevant. This human-like quality in responses fosters a more conversational and immersive user experience, ultimately contributing to higher user satisfaction and retention rates.

Mosque Locations

The Mosque Locations feature, while straightforward in its implementation, holds significant value within the app. This addition stemmed from user feedback during app testing sessions, where users expressed the need for easy access to nearby mosques. Despite its simplicity, this feature didn't detract from the development timeline for more critical features like prayer times and events.

Originally, the feature included only the Northumbria Mosques on campus. However, it was later expanded to encompass other mosques in Newcastle accessible by public transport. This expansion aligns with the app's objective of assisting international students and faculty at Northumbria with their religious obligations.

The frontend implementation of this feature is centred around a method called 'openLocation' (See Figure 55). This method facilitates the seamless navigation to multiple mosques with a simple button tap, leveraging Google Maps. By encoding the mosque's address and constructing a URL with the appropriate query parameters, users are redirected to the correct location within the Google Maps app.

```
const Locations = () => {
  // Function to open Google Maps at the specified location
  const openLocation = (address) => {
    const encodedAddress = encodeURIComponent(address);
    const url = `https://www.google.com/maps/search/?api=1&query=${encodedAddress}`;
    Linking.openURL(url).catch(err => console.error("Couldn't load page", err));
  };
```

Figure 55. 'openLocation' method which helps create the URL that is then used to redirect users to google maps for their chosen Mosque.

This innovative approach sets the app apart, particularly in comparison to other religious apps for students. While many such apps typically include only affiliated mosques, limiting user's options, the Northumbria ISOC app enables users to discover and access multiple mosques (See Figure 56). Moving forward, this feature could be further enhanced by incorporating a functionality that displays the distance to the nearest mosque directly within the app, allowing users to make informed decisions based on proximity. This addition would refine user convenience by enabling choices that suit immediate needs, thus enhancing the overall user experience and aligning with the app's commitment to user-centric innovations.

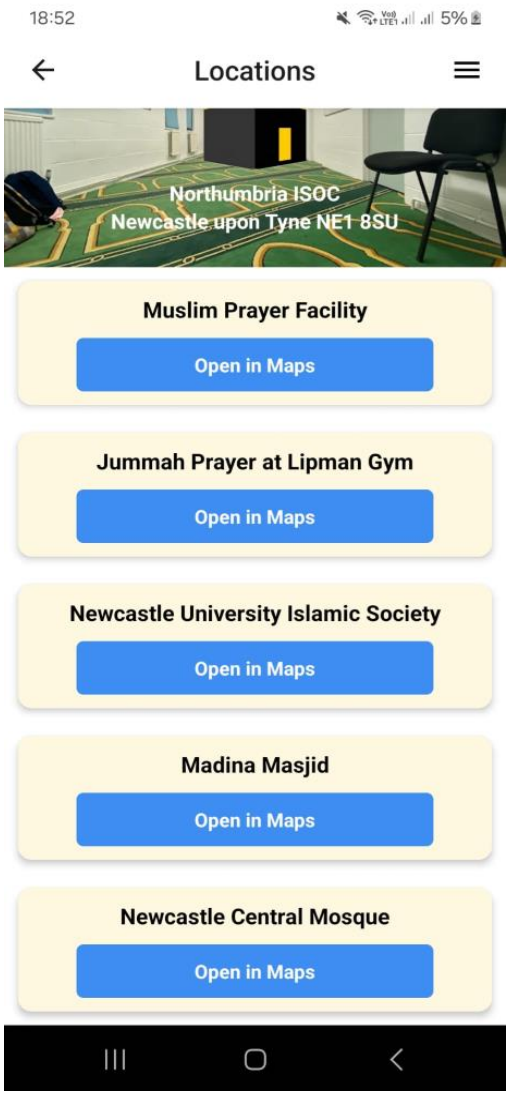


Figure 56. Clear Mosque names with clear buttons that would redirect users to google maps when they click on their preferred Mosque.

Athkar

The 'Athkar' feature, a cornerstone of the app, provides users with spiritually enriching phrases segmented into morning, evening, and before-sleep categories to facilitate daily remembrance practices tailored to different times of the day. This feature was developed after thorough research indicated that Athkar apps were among the 3 most downloaded apps when it comes to the Muslim community.

The backend of the 'Athkar' feature is managed by the ‘AzkarController’, which is designed to handle requests efficiently and flexibly through three distinct endpoints: ‘morningAzkar’, ‘eveningAzkar’, and ‘nightAzkar’. Each endpoint caters to a specific category of remembrance phrases, allowing users to access tailored content that corresponds to different times of the day.

This controller utilises Laravel's Eloquent ORM to construct dynamic queries based on the category of Azkar, ensuring that data retrieval is both fast and resource efficient. If a user specifies an ‘Azkar_id’ through a query parameter, the controller refines the data retrieval process to return only the requested Azkar, thereby supporting personalized user interactions and minimizing data transfer.

For instance, when handling a request for morning Azkar, the controller filters the Azkars specifically categorized under 'morning' and, if provided, further narrows down the results by the specified

Azkar_id. This approach not only enhances the speed and relevance of the responses but also ensures that the system can scale efficiently with increasing user demands (See Figure 57).

```php
// Start the query
$query = Azkar::select('Azkar_id', 'Text', 'Category', 'Translation', 'Repitition', 'Name')
->where('Category', 'morning');
// If a specific Azkar_id is requested, filter the query
if ($azkarId !== null) {
    $query->where('Azkar_id', $azkarId);
}

$azkar = $query->get();
```

Figure 57. Athkar query selector which retrieves data based on Azkar_id if provided.

By structuring the API to selectively retrieve data, the backend supports a responsive and engaging user experience, making the app both powerful and easy to navigate. This methodical backend setup ensures that the app remains responsive and efficient, even as the database grows and user interaction patterns become more complex.

On the frontend, the 'Morning Azkar' module exemplifies the thoughtful integration of functionality and aesthetics. It employs the ‘PagerView’ component to offer a seamless, swipeable interface that mimics a natural reading experience, thus respecting the cultural context of the Athkar. The visual design subtly shifts across different sections of the app, light blue themes in the morning echo the serenity of dawn, while darker tones towards the evening reflect the onset of dusk, enhancing the user's emotional and spiritual engagement.

The interaction is made intuitive with the ‘handlePress’ function, which not only tracks the repetitions of each Athkar but also smartly navigates through the collection. A dynamic progress bar visually represents the user's advancement through the Athkar phrases, offering instant feedback on their progress and fostering a sense of accomplishment upon completion (See Figure 58).

```javascript
const handlePress = (repitition) => {
  if (currentCount < repitition) {
    setCurrentCount(currentCount + 1);
  } else if (currentPage < azkarData.length - 1) {
    const nextPage = currentPage + 1;
    setCurrentPage(nextPage);
    setCurrentCount(1);
    pagerRef.current?.setPage(nextPage);
  } else {
    setIsCompleted(true);
  }
};
```

Figure 58. This snippet showcases how user interactions are managed, counting repetitions and transitioning between different Athkar, thereby enhancing the usability of the app.

Looking forward, integrating features such as real-time updates for new Athkar entries and customisable reminders could further personalise the experience. Implementing machine learning algorithms to suggest Athkar based on user behaviour and times of high engagement could also be explored to make the app not only a tool of practice but also a companion in spiritual growth.

The 'Athkar' feature's technical excellence, coupled with its profound understanding of user needs and cultural contexts, sets it apart in digital religious practice. It not only serves the functional needs of the users but also enriches their spiritual lives, demonstrating a high standard of software development and user experience design.

Testing

Ask me chatbot

The "Ask Me Chatbot" feature, leveraging the OpenAI API, presented a streamlined approach to integration, focusing primarily on ensuring the chatbot responded effectively and efficiently to user inquiries. Unlike more complex features, this component did not necessitate extensive backend unit testing but was rigorously evaluated through integration and usability testing to optimise user interaction.

Integration testing primarily utilised Postman to evaluate the OpenAI API's performance. This testing phase was critical to assess the response time and the accuracy of the chatbot's replies (See Figure 59). Meticulous checking in the response format and speed were implemented to ensure the chatbot could handle a realistic flow of user queries without delays, thereby maintaining a seamless user experience.

- **Response Time Measurement:** Response times were systematically recorded to ensure they met our performance criteria, providing feedback within a few seconds to keep user engagement high.
- **Accuracy and Relevance:** Tests were designed to verify that responses were not only prompt but also relevant and accurate, enhancing the utility of the chatbot in real-world applications.

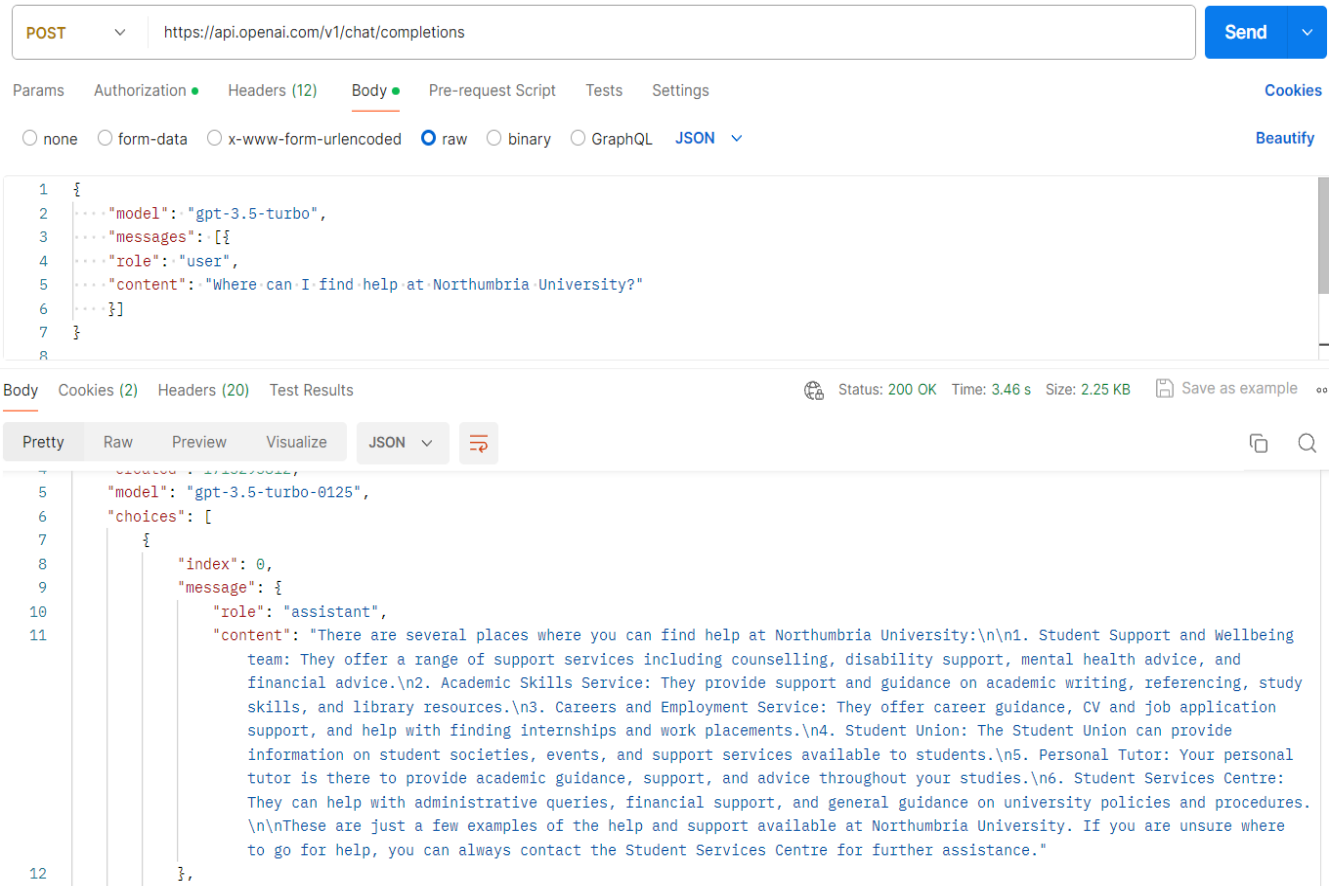


Figure 59. Chatbot example question and response where a question was asked "Where can I find help at Northumbria University, where the chatbot prompted a very good response that would be able to help users get the support they need.

Usability testing involved a diverse group of users interacting with the chatbot across various scenarios. Participants were instructed to pose questions typical of their daily university and community-related needs to assess how well the chatbot managed real-life queries.

- **User Engagement:** During sessions, users engaged with the chatbot, asking a range of questions from simple queries about university services to more complex requests related to Islamic practices.
- **Think Aloud Protocol:** Participants were also encouraged to verbalise their thoughts during the interaction, providing real-time insights into their experience. This method helped pinpoint any confusion or miscommunication, allowing us to refine the chatbot's conversational algorithms.

Participants consistently noted the chatbot's effectiveness in delivering detailed, conversational responses, which was a decisive factor in choosing OpenAI's API over more factually oriented APIs like Gemini. This preference underscored the importance of a conversational tone in enhancing user satisfaction and engagement.

The chatbot was highly praised for its conversational abilities, particularly its capacity to extend beyond direct answers by providing additional, contextually relevant information (See Figure 60). Feedback highlighted a key enhancement opportunity: the ability to save chat histories. Users expressed that saving their chats would allow them to revisit valuable information and continue conversations later. Implementing this feature could significantly boost the chatbot's utility, making it an even more indispensable resource for users seeking continuous interaction and reference.

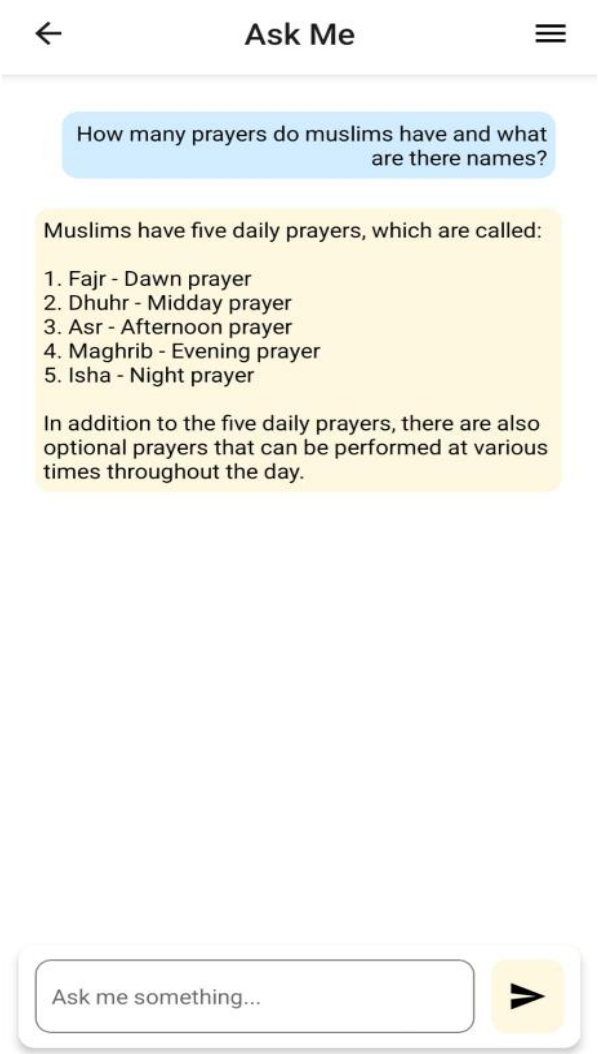


Figure 60. User question during usability testing which showcases the chatbot giving appropriate responses and going beyond by responding with required information and extra information for extra knowledge.

Mosque Locations

For the "Mosque Locations" feature of the Northumbria ISOC app, usability testing was integral in assessing the user interface and functionality. This feature allows users to access directions to various mosques around Newcastle via Google Maps, directly from the app. During the testing phase, a diverse group of users including students and faculty from different backgrounds were asked to locate mosques to evaluate the navigation's intuitiveness and the map integration's effectiveness.

Feedback from this phase was predominantly positive, highlighting the feature's practicality. Users appreciated the simplicity with which they could find mosque locations, which proved especially beneficial for those unfamiliar with the city. One user commented, "It’s really helpful to have quick access to mosque locations, especially when I'm in a part of town I'm not familiar with." This kind of feedback validated the feature's utility in helping users navigate new areas smoothly.

However, users also suggested practical enhancements. A frequent suggestion was the addition of distance information to each mosque listing, allowing users to choose the closest mosque based on their current location. "Knowing how far each mosque is would really help me decide the best one to go to depending on my schedule," mentioned one of the participants. This feature was seen as a valuable enhancement that would make the app more useful for planning visits, particularly during busy periods like Friday prayers.

Additionally, while users found the clickable buttons that launched directions in Google Maps convenient, some recommended integrating real-time traffic updates to assist in planning their journeys more effectively. "If it could show how busy the roads are, it would help me plan better and arrive on time," suggested another user.

These insights were instrumental in refining the feature. They not only confirmed its functional adequacy but also opened avenues for further enhancements, making the Mosque Locations tool not just a functional necessity but a considerate addition to the user's mobile toolkit. The usability testing phase, therefore, not only underscored the feature's current successes but also mapped out potential improvements to enrich the user experience.

Athkar

The Athkar feature underwent extensive testing to ensure its functionality and user satisfaction. Initially, unit tests were developed to validate the retrieval processes for the different Athkar sections morning, evening, and before sleep using mock data that mimicked the expected outputs. These tests successfully confirmed that each function could accurately pull the appropriate data, including athkar names and their repetitions, from the database (See Figure 61).

```
    $expectedAzkar = [
        [
            'Azkar_id' => 1,
            'Text' => 'أَعُوْذُ بِاللّٰهِ مِنَ الشَّيْطَانِ الرَّجِيْمِ...',
            'Category' => 'morning',
            'Translation' => 'I seek the protection of Allah from the accursed Shayṭān...',
            'Repitition' => 1,
            'Name' => 'Ayat al-Kursi: The Greatest Protection'
        ],
        [
            'Azkar_id' => 2,
            'Text' => 'Second text...',
            'Category' => 'morning',
            'Translation' => 'Second translation...',
            'Repitition' => 2,
            'Name' => 'Second name'
        ]
    ];

    // Create an instance of the controller
    $controller = new AzkarController();

    // Create a request instance with necessary parameters
    $request = Request::create('/azkar/morning', 'GET');

    // Call the method to test
    $response = $controller->morningAzkar($request);

    // Convert the actual result to an array
    $actualAzkar = $response->original->toArray();

    // Assert that the method returns the expected result
    $this->assertEquals($expectedAzkar, array_slice($actualAzkar, 0, count($expectedAzkar)));
}
```

Figure 61. unit test for the morning azkar which showcases how the expected data which mimics real azkar data is used to assess if the expected outcome would be achieved, similar tests for both evening and before sleep azkar were carried out.

Following the unit testing, integration tests were conducted with Postman to verify the correctness of API endpoints. These tests focused on ensuring that data was retrieved in the correct format and assessed the response times to maintain performance standards. The results indicated that the API was efficient, with fast response times that met the project specifications. It was also confirmed that the

APIs were secure, allowing only GET requests to prevent unauthorised modifications to the athkar data.

In the usability testing phase, three users were selected to interact with the different Athkar sections. Their task was to navigate through the app and engage with the content, providing feedback on their experience. The participants reported that the athkar were comprehensive and accessible. One user remarked, "The layout is intuitive, making it easy to find and engage with the athkar throughout the day." This feedback highlighted the successful integration of the Athkar content into the user interface.

However, valuable suggestions emerged during these sessions. One user suggested implementing a counter for athkar that required multiple repetitions, explaining, "It's helpful to have a counter for athkar like those that require 33 repetitions. It helps keep track and stay focused, especially when you're trying to balance prayer with a busy schedule." Another user proposed the idea of a progress bar, noting, "A progress bar would be great for us to see how much we've completed and what's left. It would make it easier to come back and continue where I left off if I get interrupted." These insights were instrumental in identifying areas for enhancement, emphasising the need for features that support the user's prayer practice in a dynamic, daily context.

By integrating feedback directly from the usability tests, the Athkar feature was refined to better cater to the needs of Northumbria University's Muslim community, ensuring that it not only serves their spiritual needs effectively but also respects their time and daily routines (See Figure 62).

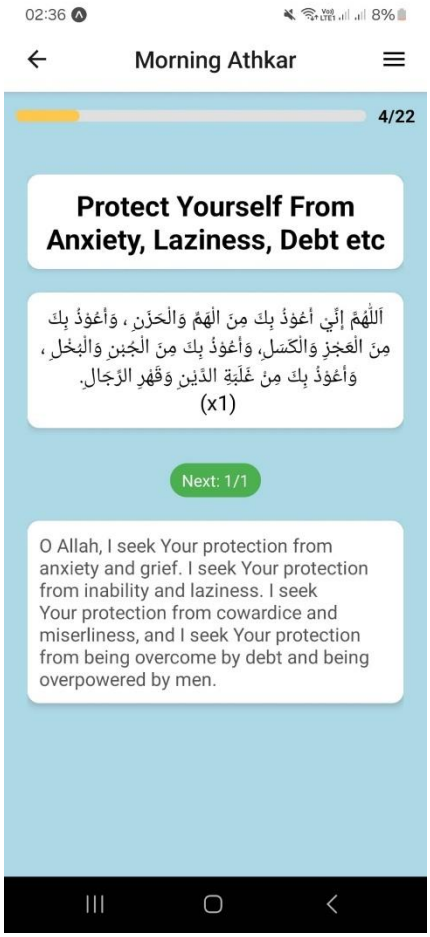


Figure 62. User feedback was implemented as visible by the progress bar up top and the counter in the middle both enriching the overall user experience.

Appendix D: Feedback Loops and Usability testing

Prayer Times

User ID	Prayer Times Task	Rating (1-5)	Time to complete task from app launch (seconds)	Iteration of feature	Feedback	Insight gained
1	Find the next prayer time	5	3	2	"It was very easy to find the next prayer by using the countdown and the highlighter."	N/A
2	Find the next prayer time	3	7	1	"I found it as soon as I opened the app on the homepage, but I feel like the countdown is a bit too small to read."	Font size is too small, larger font needed.
3	Find the last prayer time for today	4	4	1	"The prayer times are all clear to see, this makes it easy for me to schedule my busy day."	Talked about adding a qibla direction so they can pray wherever they are and get back on with their day.
4	Find the first prayer time for today	4	5	2	"The prayer times were all displayed in a grid format which makes it easy to identify which time is for which prayer."	Insight on adding a prayer calendar for tracking important days of the month.
5	Find the next prayer time	5	4	3	"I loved the blue highlighter it makes it super easy to know which prayer is next, with the countdown it's perfect."	N/A

Quran

User ID	Quran task	Rating (1-5)	Time to complete task from app launch (seconds)	Iteration of feature	Feedback	Insight gained

1	Find page 3 of the Quran	4	10	3	“I found the Quran section quickly from the menu then navigated to the page tab and that’s where I found the 3rd page of the Quran .”	Final iteration is meeting user needs.
2	Find page 61 of the Quran	4	7	3	“The search box helped me reach the page quickly.”	The search box implemented increased performance
3	Find page 300 of the Quran	3	16	2	“I kept scrolling till I got to page 300 which took time.”	Scrolling was an issue when pages were further down suggesting the need for a direct way to get there.
4	Find page 522 of the Quran	2	18	2	“Reaching page 522 was way too hard by just scrolling.”	Emphasised the need for a search box to reach there quicker
5	Find the Quran section	5	5	1	“The menu was placed perfectly for me to find all the functionalities of the app.”	Menu was easily accessible and helps show other features of the app well
6	Find the Quran section	5	8	1	“The Quran section was clearly labelled in the Menu, so I was able to get there quickly.”	Quran feature was well labelled in the Menu, so users got there quickly
7	Find surah 4	5	10	2	“As soon as I opened the Quran feature from the Menu, I found the Surah.”	Surah’s were displayed in a clear manner
8	Find surah 42	4	12	3	“The Search box added was helpful for directly entering the number of the surah to find it.”	Search box helped navigate around the surahs easily
9	Find surah 100	5	11	3	“I used the Menu to go to the Quran section then I found a search box where I tried to write 100 and I was taken to the surah number 100.”	Navigation using the search box was seamless and worked as expected.

10	Find juz 16	5	12	2	“The Juz tab made it easy to find the exact juz I was after.”	Design change helped from previous dropdown design to new tabbed layout.
11	Find juz 24	5	9	3	“Using the search box in the Juz tab allowed me to finish the task quickly.”	Search box was beneficial and illustrated how it’s helpful for users.
12	Read through page 2	5	30	2	“Text font is appropriate for the Quran.”	User ensured authentic text and added notes about the font style and size
13	Read through page 18	4	44	2	“The text is perfect, its all lined up with Quran.com text.”	Authenticated the text to ensure reliability
14	Read through page 299	5	68	3	“Spacing, navigation and font are all nicely put together for this section.”	Reading experience was good especially swiping for navigating like a book.

Events Calendar

User ID	Event tasks	Rating (1-5)	Time to complete task from events page (seconds)	Iteration of feature	Feedback	Insight gained
1	Create an event	5	53	1	“It was so easy with the add button and simply inserting all the necessary information.”	Adding events was setup as users expected.
1	Edit an event	3	30	1	“I had to put in all the information again after creating the event.”	Editing API had a few issues, when users clicked edit, the old data wouldn’t show up, meant testing was needed again and a second iteration was made.
2	Create an event	5	65	2	“Straight forward, intuitive add button with	Positive feedback specifically mentioned the

					good input descriptions like ‘occurrence’, the placeholder text makes it easy to know what type of information to type there.	placeholder text that made it easier to understand what the specific box was for.
2	Edit an event	5	15	2	“I quickly edited the title and updated the event, nice and easy process.”	New iteration with edit API working as expected, user found it easy to edit an event
2	Delete an event	5	2	2	“Tap of a button and it was gone, really enjoyed the experience.”	Delete API worked seamlessly user suggested changing the buttons from text to icons, which was implemented for a richer user experience
3	Where is the location of the upcoming event	5	5	2	“The location was coloured green, I clicked on it, and it took me to google maps to the specific location placed there, I liked that feature.”	Viewing events from a regular user’s perspective was straightforward, positive feedback on the location redirection, suggested it was a good addition to the feature.