BulSU Consult Hub: A Student Consultation Scheduler Republic of the Philippines

Aribal, Marcel James V[1], Banderado, Lorenz L [2]., Zapanta, Karen Kei A[3]., Gatdula, Ma. Magdalena V.[4]

Graduate School, Bulacan State University, City of Malolos, Bulacan, Philippines

DOI: https://doi.org/10.47772/IJRISS.2026.100900800

Received: 02 October 2026; Accepted: 12 October 2026; Published: 24 October 2026

ABSTRACT

BulSU Consult Hub: A Student Consultation Scheduler was designed to lessen manual coordination and scheduling conflicts for academic consultation experienced by graduate school students and faculty of the Bulacan State University. The project aimed to develop a web application that lets professors manage their availability, lets students book appointments, and handles approvals, rescheduling, automated notifications, and a consultation history log. A developmental software-engineering approach guided by the Agile model was applied. Project planning and feature identification were completed in Phase 1; requirements analysis, prioritization, SRS preparation, and interface wireframing were completed in phase 2; MVP and additional module development with preliminary testing were documented in Phase 3; and module integration, system testing, defect correction, and compliance review were completed in Phase 4. The prototype used React with Tailwind CSS for the client interface and Supabase as the backend layer. Functional, integration, and system tests examined navigation, validation, data creation, retrieval, updating, deletion, and error handling. Based on 16 recorded system test cases, 11 passed initially, while 5 required correction and retesting. After corrective actions, 14 of the 16 test cases passed, and the remaining two, graduated-student account validation and session-data persistence, remained unresolved. The findings indicate only that the implemented prototype successfully supports institutional authentication, availability and program-scoped scheduling, conflict-free booking, and role-based data access as specified in the SRS, within the scope and testing conditions applied in this study. Further work should address automated enrollment-status verification and session-resilient form data recovery before the system is considered ready for full deployment.

Keywords: Academic Consultation Scheduling, Agile Software Development, Software Engineering, Software Testing, Schedule Conflict Locking

INTRODUCTION

Software engineering is the systematic approach of converting an idea into a functional, reliable system. Programming alone is not enough as requirement analysis, system designing, development, testing, and maintenance should be incorporated in the process to ensure that a system is functional, secure, and responsive to the actual needs of its users. In this project, the group applied software engineering principles to turn the approved software concept into a tested, functional prototype through four linked course activities.

Academic consultations between students and professors at the Bulacan State University's Graduate School Department are mostly arranged through channels like Messenger chats, emails, or hallway requests. This setup creates several difficulties; students cannot see the actual availability of their professor before requesting a consultation, back-and-forth exchange of messages are needed to come up with a schedule acceptable for both, and no centralized way to confirm and track the status of requests. These can result in missed appointments, overlapping commitments, and delays in academic guidance, especially for professors handling students from different programs all at once. Therefore, a need for a centralized platform that allows students who seek academic consultations can view the availability of their professors, request consultation, receive real-time status updates, while giving full control to the professor to manage their own schedules, and approve requests without relying too much on manual coordination.

The system aims to design, develop, integrate, and test the BulSU Consult Hub, a web-based scheduler that modernizes how consultations are booked, approved, and managed between graduate school students and professors at BulSU Main Campus. Specifically, it aims to: (1) design and implement an authentication module that that limits system access to verified BulSU accounts only; (2) develop a module that allows the professors to set their availability, choose preferred consultation modes, customized availability by program, and manage all requests; (3) develop a consultation booking module that allows students to browse professors' availability, and request consultation with a stated purpose, concern type, and other details; (4) implement an automated schedule-conflict locking module to prevent double-booking and overlapping consultation schedules; (5) implement a cancellation and rescheduling policy module that enforces a minimum of one day advance notice for approved appointments; (5) retrieve past, pending, and incoming appointment records; (6) integrate automated notifications and real-time dashboard synchronization for students and professors to receive timely updates without manual refreshing; and (7) conduct module-level, integration, and system-level testing of the prototype to verify its satisfaction based on the approved SRS and documents.

The prototype covers the design, development, and testing of the BulSU Consult Hub as a web-based application for graduate school students and professors under the Graduate School Department of Bulacan State University Malolos Campus. The system includes institutional sign-in restricted to @ms.bulsu.edu.ph accounts, instructor schedule and availability management, student consultation booking, an approval or decline workflow, automated schedule-conflict prevention, a one-day advance cancellation or rescheduling policy, automated in-system and email notifications, and real-time dashboard updates. The development does not cover a mobile application, a built-in messaging feature, video-conferencing engine for approved online appointments, other departments aside from the Graduate School and a multi-campus rollout. The system was developed within a fixed four-week timeframe as part of MS613 course requirement, which constrained the depth of features. Testing was limited to functional, integration, and system test cases executed by the development team. This study may benefit: (1) Graduate School Students by giving them a transparent, self-service way to view their professors available consultation schedules and track their requests in real time; (2) Graduate School Faculty by having control over their consultation schedules, ability to approve, decline, and filter requests by program, organize queue that helps in reducing missed and duplicated consultation appointments; (3) the Development Team as they have gained practical and hands-on experience in applying software engineering and web development competencies; and (4) the Future Researchers and Developers as the used methodology, SRS, testing, and the identified limitations may serve as reference for developing similar scheduling system.

Most software engineering studies consistently frame Agile SDLC as their top choice for projects that need to accommodate evolving requirements within a fixed timeline. Its incremental approach can handle shifting stakeholder requirements better than models like Waterfall, particularly in team-based, time-constrained projects (Fagarasan et al., 2021). Agile must also be paired with disciplined testing and review after every sprint to improve the system outcome (Saarikallio and Tyrväinen, 2023). Also, with this, planning requirement prioritization usually use "MoSCoW" technique, a formalized practice which proved that separating "Must Have" functionality from deferrable features in resource-constrained and sprint-based projects is effective (Kravchenko et al., 2022). However, a revisitation of the "MoSCoW" rules as cautions that Must-Have ratios can undermine its value (Miranda, 2023). These studies support the idea that Agile's iterative and feedback-driven approach is well-suited to small, capstone-scale teams.

A systematic review of web-based medical appointments found reduced no-show rates and improved patient-reported convenience upon scheduling online (Zhao et al., 2017). In academic setting, a web-based appointment system built for a university president's office scored highly on both usability and technology acceptance. This suggests that digitizing a walk-in based appointment process can improve usability and usefulness (Ampuan & Deleña, 2022). Also, a dedicated Student Appointment and Consultation System (SAPCoS) developed for a Malaysian university reported that both students and lecturers found the application helpful for planning, tracking, and receiving notifications about consultation appointments (Hashim et al., 2019). A similar study at a university in United Kingdom, where a centrally managed booking system was introduced for academic-skills appointments, reportedly made staff-student scheduling more convenient to coordinate (Morris and McDermott, 2020). These studies proved that digitization of appointment-booking leads to measurable gains in convenience and cuts down manual coordination, which drives the system's development.

For the technical stack of the system, Next.js's server-side rendering solves the initial load problem by generating pages on the server before they were ever sent to the browser, which works well for dashboard-heavy systems like this consultation scheduler (Jartarghar et al., 2022). On the backend, Supabase relies on PostgreSQL's logical replication to send INSERT, UPDATE, and DELETE events to connected clients through WebSockets. It also uses Row-Level Security (RLS) policies to control access to these real-time updates. This supports the BulSU Consult Hub's almost real-time dashboard synchronization. PostgreSQL also provides stronger relational integrity and SQL-based security than other NoSQL alternatives (Netguru, 2025), which suitable for the system's need since it depends on strict database rules. On the frontend, Tailwind CSS's utility-first approach can help reduce repetitive CSS and keep layouts consistent and responsive across different devices (Azhariyah et al., 2022).

On the system quality side, software architecture decisions should be planned carefully and documented based on the quality requirements they are intended to meet (Bass et al., 2021). Software development should also follow an organized life cycle process covering activities such as requirements, development, and verification (ISO/IEC/IEEE, 2026). For the user interface, human-centered design emphasizes designing systems around users, their tasks, and their actual environment of use (ISO, 2019). Usability engineering and user-interface design also recommend considering usability throughout the development process and creating interfaces that are easy for users to understand and operate (Nielsen, 1994; Shneiderman et al., 2016). User acceptance can also be influenced by how useful and easy to use users perceive a system to be (Davis, 1989). Good code quality can be supported through clear coding practices, proper structure, and regular refactoring (Martin, 2008; Fowler, 2018). Design patterns may also be used as established solutions to common software design problems (Gamma, et al., 1995).

For software delivery, continuous integration and deployment practices can support faster and more reliable software delivery (Forsgren, et al., 2018; Humble and Farley, 2010). Software testing should be conducted at different levels and should include clear criteria for determining whether the system meets its requirements. Verification and validation practices help determine whether the developed system satisfies its requirements and intended use (IEEE, 2024).

Since BulSU Consult Hub handles students' and faculty members' personal information, security should be considered throughout the software development life cycle. The system must comply with the Philippine Data Privacy Act of 2012, which requires the protection of personal information processed by government and private organizations (National Privacy Commission, 2012). A case study of RA 10173 implementation within a Philippine higher-education institution (Remigio, 2024) found that universities tend to fall short in enforcing role-based access and third-party data sharing. The BulSU Consult Hub is designed to avoid this gap by having role-based RLS policies.

Existing studies also backs up the choice of an Agile SDLC, paired with sprint-based testing and MoSCoW-based requirement prioritization, is best for time-constrained projects with evolving requirements (Fagarasan et al., 2021; Saarikallio and Tyrväinen, 2023; Kravchenko et al., 2022), though Miranda (2023) points out that MoSCoW only works if the Must-Have ratio is applied with discipline rather than arbitrarily. Studies on appointment-booking systems consistently show that digitizing scheduling lessen coordination overhead and improves usability, convenience, and satisfaction, whether in healthcare (Zhao, et al., 2017), university administrative offices (Ampuan and Deleña, 2022), student-lecturer consultation platforms (Hashim et al., 2019), or academic-skills scheduling (Morris and McDermott, 2020). Supporting standards further emphasize deliberate architecture-to-quality alignment (Bass et al., 2021), documented lifecycle processes (ISO/IEC/IEEE, 2026), human-centered and usable interface design (ISO, 2019; Nielsen, 1994; Shneiderman, et al., 2016; Davis, 1989), sound coding practices (Martin, 2008; Fowler, 2018; Gamma, et al., 1995), and verification through CI/CD and formal testing criteria (Forsgren, et al., 2018; Humble and Farley, 2010; IEEE, 2024).

Comparable systems commonly provide digitized booking, notifications, and usability gains over manual or walk-in processes (Hashim et al., 2019; Ampuan and Deleña, 2022; Morris and McDermott, 2020), but limitations remain at the institutional level: Remigio's (2024) case study of RA 10173 implementation in a Philippine higher education institution found that universities most commonly fall short in role-based access

enforcement and third-party data sharing controls, indicating that Data Privacy Act compliance in Philippine HEIs is frequently underenforced at the technical access-control layer, regardless of whether a system's core scheduling features are otherwise usable and effective. This project addresses the gap directly by implementing a role-based consultation scheduler with Supabase's Row-Level Security policies at the database layer, and by embedding strict relational constraints, such as allowing only one approved consultation per timeslot into the PostgreSQL schema. This means that the system is designed to enforce technical control in compliance to the Data Privacy Act of 2012, directly responding to the enforcement gap that Remigio (2024) identified as common among Philippine HEIs.

Definition of Terms. Functional requirement – a service the system must provide. Non-functional requirement – a quality or constraint on how the system should performs. Module testing – independent testing of one feature. Integration testing – testing how connected modules interact with each other. System testing – evaluating the complete system against the specified requirements. Single Sign-On (SSO) – an authentication method allowing a user to log in once through a trusted identity provider to access a system. Row-Level Security (RLS) – a database-level access-control mechanism that determines which records a user can read or modify. Schedule conflict locking – a mechanism that automatically prevents a time slot from being booked by more than one confirmed party once a request for it has been approved.

METHODOLOGY

The project utilized Agile model as a developmental software engineering approach, organized into four-week sprints. This approach was chosen because it fits the system’s technology, structure, and team dynamics well, and it kept the team accountable in delivering a working increment after every sprint instead of a single deliverable at the end of four weeks.

The first phase established the project concept where the team identified the problem, system’s intended functionalities, development environment, and the task assignment of members. The second phase identified the stakeholders, SDLC model to be used, prioritized requirements, Software Requirement Specifications, system flow, and the initial wireframe. The third phase is where the planned features are converted into working modules, interface improvements done, system connections to the database, and preliminary module testing. Finally, in the fourth phase, modules were integrated, realistically tested in an end-to-end scenario, documented bugs, retested corrections, and compared the prototype with the approved SRS.

Table 1 Approved System Requirement Specifications (SRS) Mapped to Priority and Verification Method

ID	Approved requirement	Priority	Verification
FR-01	The system shall allow users to sign in using Microsoft authentication.	Must	Functional/system test
FR-02	The system shall create or retrieve user profiles after login.	Must	Data-operation test
FR-03	The system shall allow students to complete required profile information.	Should	Functional/input-validation test
FR-04	The system shall allow instructors to create consultation availability windows.	Must	Functional/system test
FR-05	The system shall allow instructors to set consultation mode: F2F, Online, or Both.	Must	Functional/system test
FR-06	The system shall allow instructors to limit availability by program.	Should	Functional/system test
FR-07	The system shall show only matching availability to students based on program.	Should	Data-filtering test
FR-08	The system shall allow students to request a consultation within available time windows.	Must	Functional/system test
FR-09	The system shall require concern type and concern details when requesting.	Must	Input-validation test
FR-10	The system shall allow instructors to approve or decline requests.	Must	Functional/integration test

FR-11	The system shall prevent overlapping approved consultation schedules.	Must	Integration/system test
FR-12	The system shall notify students when requests are approved, declined, or cancelled.	Must	Integration/system test
FR-13	The system shall notify instructors when new requests are submitted.	Must	Integration/system test
FR-14	The system shall show consultation history with search support.	Should	Functional/search test
FR-15	The system shall allow calendar invite download for approved consultations.	Should	Functional/integration test
FR-16	The system shall allow cancellation before the cutoff period only.	Must	Integration/system test
FR-17	The system shall provide dashboard update in near real time.	Should	Real-time sync/system test
NFR-01	The system shall be responsive for desktops, tablets, and mobile screens.	Should	Compatibility/Performance test
NFR-02	The system shall use role-based access control to prevent unauthorized access.	Must	Security/Performance test
NFR-03	The system shall protect database records using Supabase Row Level Security.	Must	Security/Performance test
NFR-04	The system shall prevent duplicate or conflicting approved bookings.	Must	Reliability/System test
NFR-05	The system shall maintain consultation records for accountability.	Should	Integration/System test
NFR-06	The system shall restrict access to verified institutional accounts.	Must	Security/Performance test

Table 1 shows the seventeen (17) functional requirements and six (6) non-functional requirements to outline how the system should perform considering requirements related to system's usability, performance, reliability, security, compatibility, and maintainability. These requirements are also classified based on their importance using MoSCoW rules and their means of verification.

The prototype used a three-tier client-server architecture. The presentation layer takes in user actions and displays feedback such as confirmation messages and validation errors. The application layer handles validation and business rules, including required-field checks, the schedule-conflict lock, and the 1-day cancellation/rescheduling cutoff. The data layer stores and retrieves records through a Supabase-managed PostgreSQL database, using Row Level Security policies to ensure that each user can only read or modify records tied to their own role and account.

Figure 1 Approved BulSU Consult Hub Flowchart

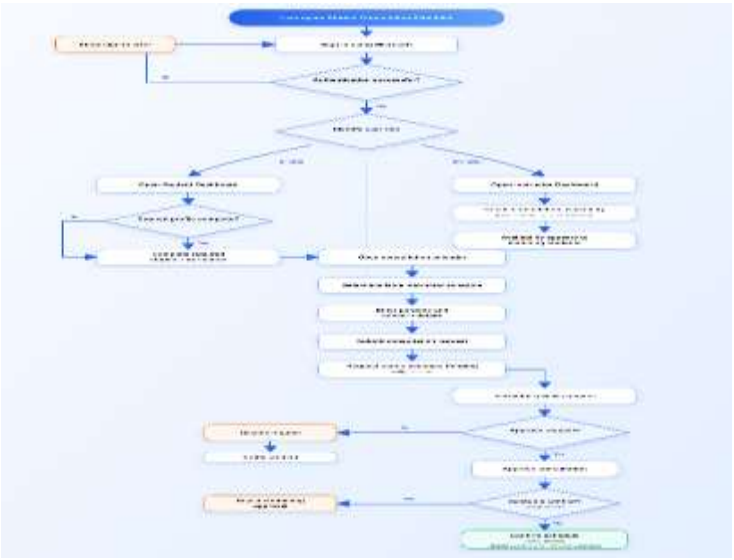


Figure 1 depicts the system flowchart. A user begins by signing in with their institutional @ms.bulsu.edu.ph account through Microsoft Entra ID, proceeds to either browsing instructor availability and submitting a consultation request (for students) or setting availability windows and reviewing pending requests (for faculty), and receives a real-time status update (Pending, Approved, or Declined) on their dashboard along with an automated email notification and, for approved consultations, a downloadable calendar invite. Access to restricted functions is controlled through Microsoft Entra domain-restricted single sign-on combined with Supabase Row Level Security policies, which together prevent non-institutional accounts from signing in and prevent authenticated users from viewing or acting on records outside their own role.

Figure 2 Entity-Relationship Diagram



Figure 2 displays the data layer and how it is implemented as a relational database in Supabase composed of six interrelated tables that together support the system workflow. A central profiles table extends Supabase's built-in authentication table with role-based user details for both students and faculty; from there, instructors create timeslots in instructor_availability, which can be scoped to multiple academic programs through a junction table, availability_programs, while students submit bookings against those slots through consultation_requests, which links each request to its requesting student, receiving instructor, and chosen time window and tracks its status through approval or decline. Last table, email_logs, records every automated notification sent for auditing. These relationships enforce data integrity and combined with role-based access policies, ensure that each user can act based on their role. This directly supports the system's core requirements for program-scoped availability, conflict-free scheduling, and automated notification.

The proposed technology stack for the BulSU Consult Hub combines a modern full-stack framework with cloud-managed backend services and deployment infrastructure.

Table 2 Technology Stack Used for BulSU Consult Hub

Tool/technology	Version	Purpose/module	Basis for selection
JavaScript/TypeScript	5.9	Programming Language	Primary programming languages used for client-side rendering and server-side application logic

Next.js	16.3.4	Framework and Library	Full-stack React framework powering the user dashboards, application routes, API routes, and server-side rendering logic
Tailwind CSS	4.3.3	UI and Styling	Utility-first CSS framework used to build and maintain a responsive, consistent interface design system
Supabase	@supabase/supabase-js	Database, Backend, and Storage	Backend-as-a-Service (BaaS) provides a PostgreSQL database, real-time database synchronization, and data storage
Microsoft Entra ID	Current MS Entra platform	Authentication and Identity	Single sign-on (SSO) and identity management provider used to restrict and verify access to @ms.bulsu.edu.ph institutional accounts
Vercel	Current Vercel platform	Hosting and Deployment Platforms	Cloud platform used to host the live application, manage continuous deployment, and deliver serverless execution
Visual Studio Code	1.136	IDE and Developer Tool	Industry-standard integrated development environment (IDE) for Next.js, Tailwind CSS, and TypeScript development

Table 2 contains the modern technology stack used for the system like the programming language, framework, IDE, database, and tools used. The system depends directly on Next.js for dashboard and server logic, Supabase for data management and real-time syncing, Microsoft Entra for user verification, and Vercel for hosting.

Upon the approval of SRS, the team started developing the user interface and the core features of the system like login and user authentication, publishing available schedules (professor-side flow), and the appointment booking features (student-side flow) to form the Minimum Viable Product (MVP). Additional features like approval/decline workflow, schedule conflict locking engine, notification alerts, record-filtering, cancellation and rescheduling of appointments were implemented as separate modules. The navigation and interface were further improved to ensure consistency and readability. Dashboard navigation was simplified by grouping related functions in a card to ensure a more user-friendly system, and that the most relevant information is visible without needing to scroll down. To ensure an organized project, the team used a shared Git repository to manage and backup the code. All new and updated modules are uploaded for review and combined with the existing features and database for easy integration.

Test cases were labeled with an ID, containing features, test input or action, expected result, actual result, status, and corrective action. Module testing was done to verify each feature independently. Integration testing checked the interface and data exchange. System testing was done to verify the whole system using realistic end-to-end scenarios covering navigation, validation, data persistence, CRUD operations, error handling, basic usability, reliability, and response time. Failed cases were entered in the bug log, assigned severity, corrected, and retested.

Test cases were labeled with an ID, containing features, test input or action, expected result, actual result, status, and corrective action. Module testing verified each feature independently, and integration testing examined how all modules communicated with one another. To check the overall functionality, system testing simulated realistic, end-to-end use cases for navigation, validation, CRUD operations, error handling, and reliability. Any test that failed was recorded in the bug log along with its severity level, then corrected and retested.

The team incorporated several safeguards to protect user data and uphold ethical standards. The data collection was kept to only what the scheduling function needed (name, institutional email, program/section, and consultation concern). User credentials were not stored directly by the system; authentication was handled entirely through Microsoft Entra ID, with access further restricted by Supabase's Row Level Security policies. Any data submitted by users was passed through client-side and server-side validation before being written to the database. Generative AI tools were used during the preparation of this documentation to assist in drafting the paper, including language refinement, and as a pair programmer; all AI-assisted content was reviewed, verified, and edited by the group members for technical accuracy, and the system's design decisions, requirements, code, and test results remain the group's own original intellectual work

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue IX September 2026

RESULTS AND DISCUSSION

The integrated BulSU Consult Hub has six (6) completed modules: User Authentication and SSO Domain Validation, Instructor Scope Filter, Student Consultation Booking, Approve/Decline Workflow, Schedule Conflict Locking Engine, and 1-Day Cancellation and Rescheduling Engine. Through the system's interface, students and professors can schedule, request, approve, and manage academic-related consultations online. Data submitted through the Consultation Request module are processed by the system's scheduling and conflict-validation logic and stored in the Supabase database, after which authorized users may retrieve and update consultation records as needed.

Figure 3 System's Landing Page

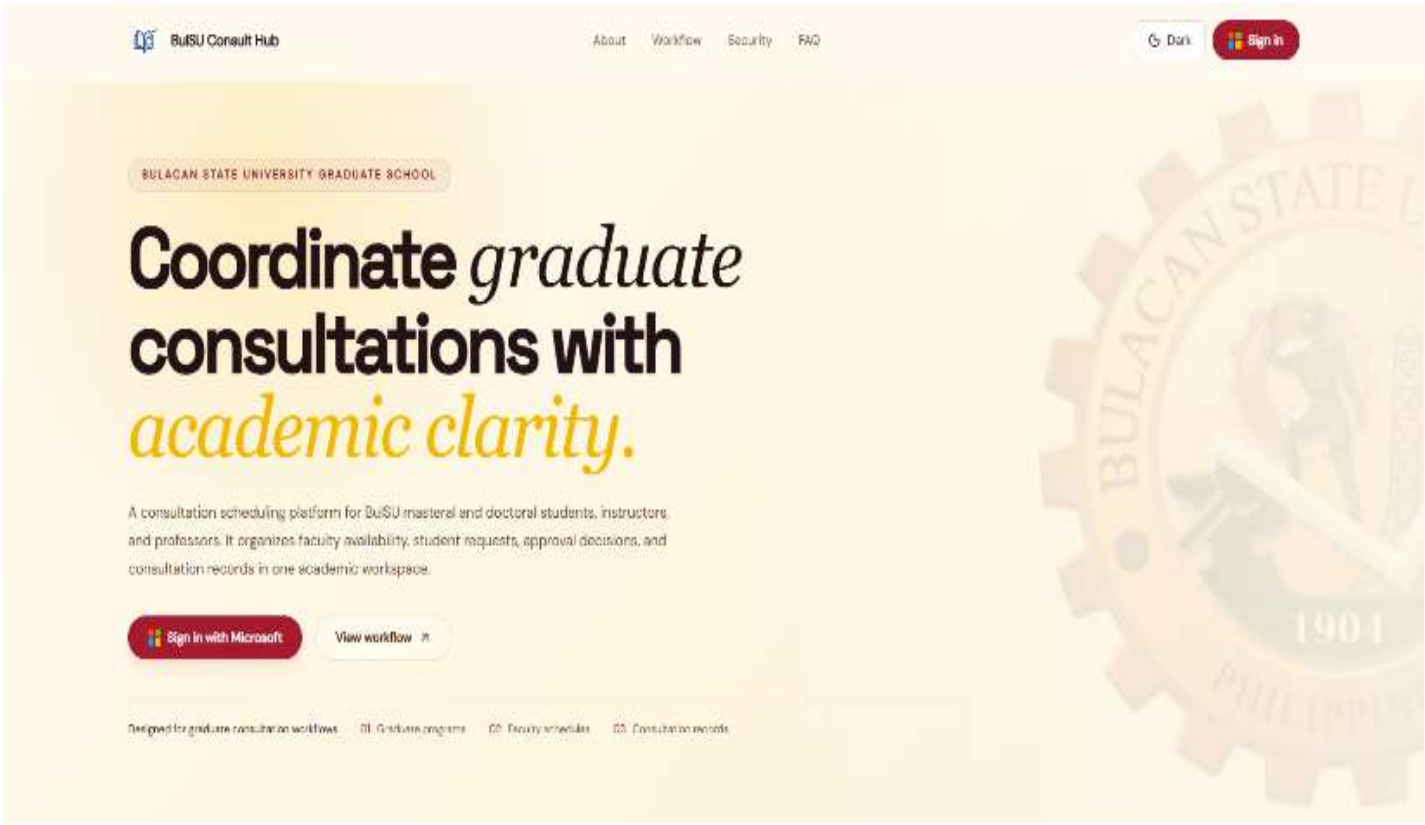


Figure 4 System's Login Page for Professors

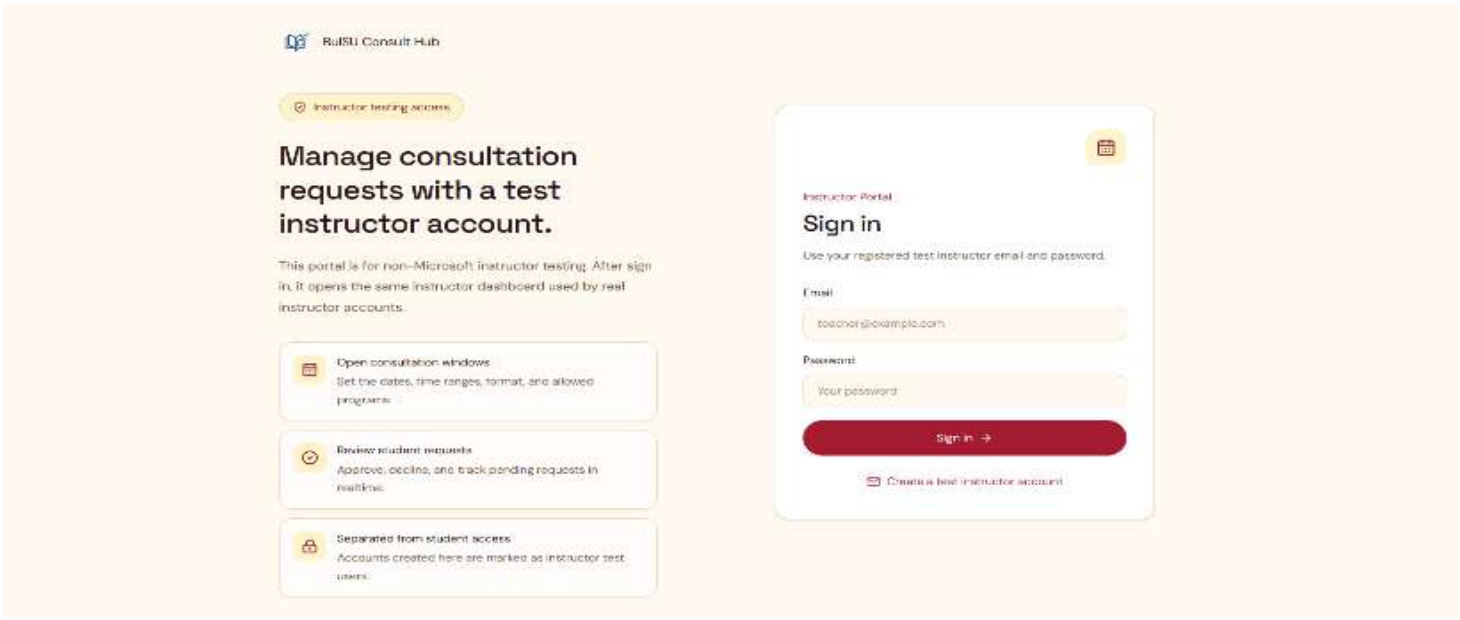


Figure 5 System's Login Page for Students

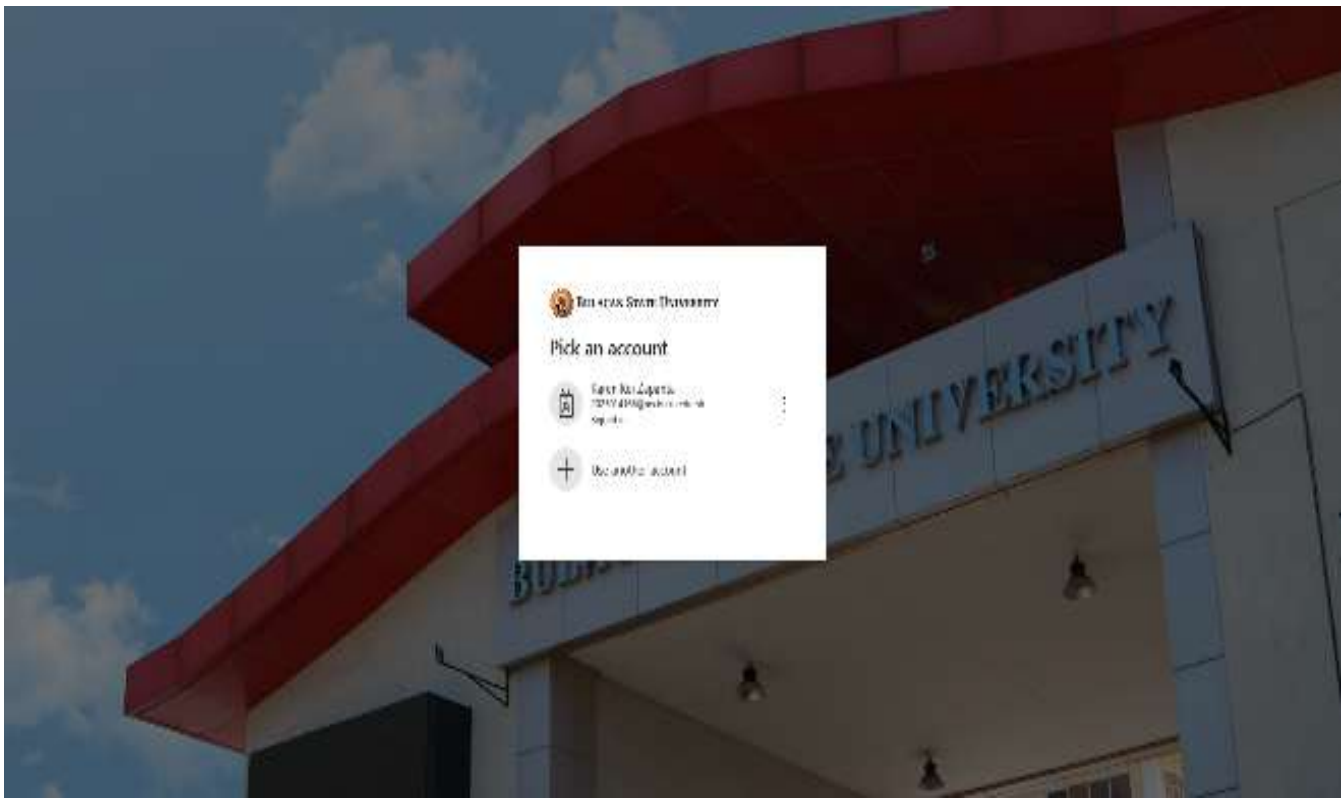


Figure 6 Professors' Dashboard

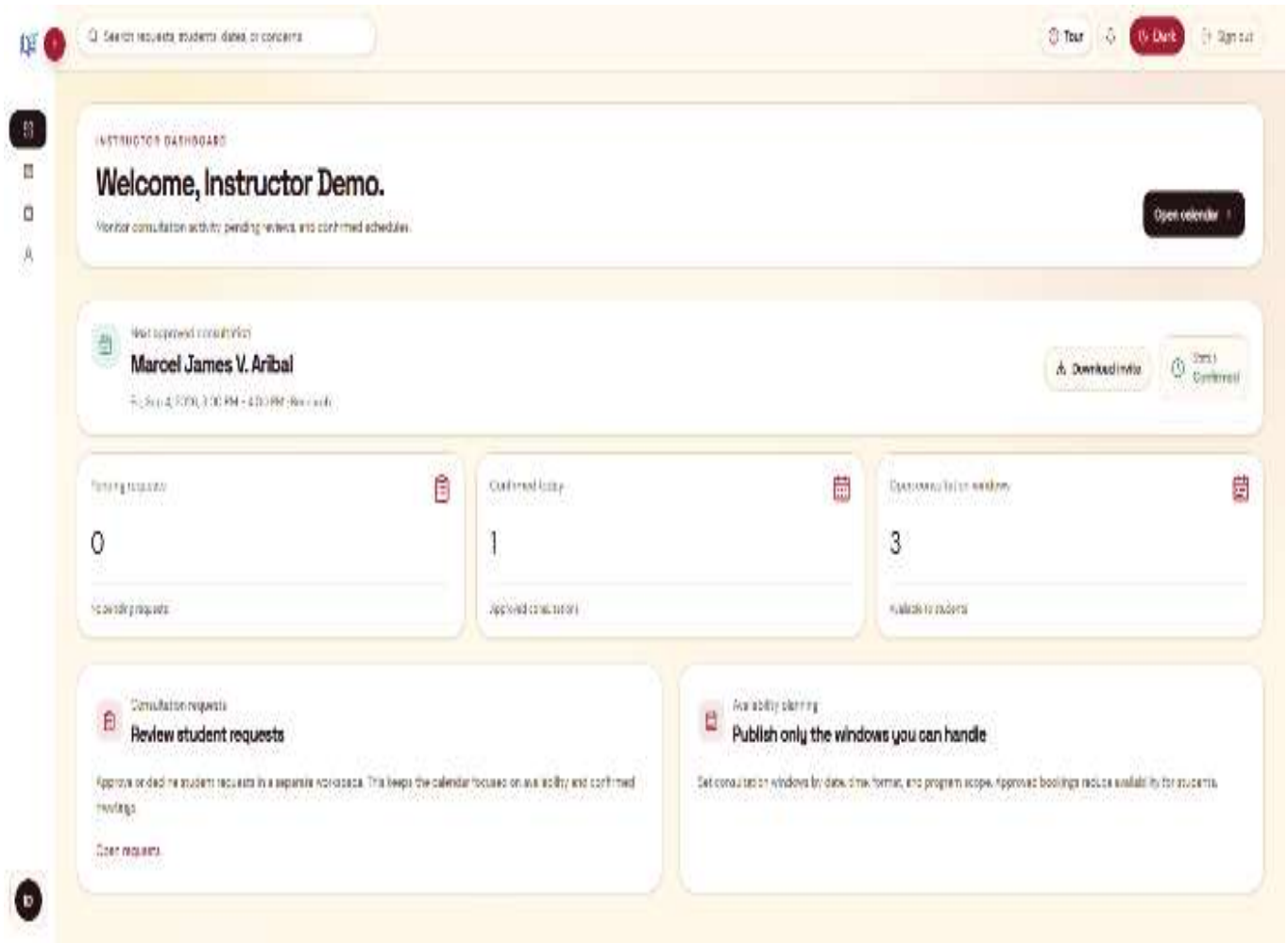


Figure 7 Students' Dashboard

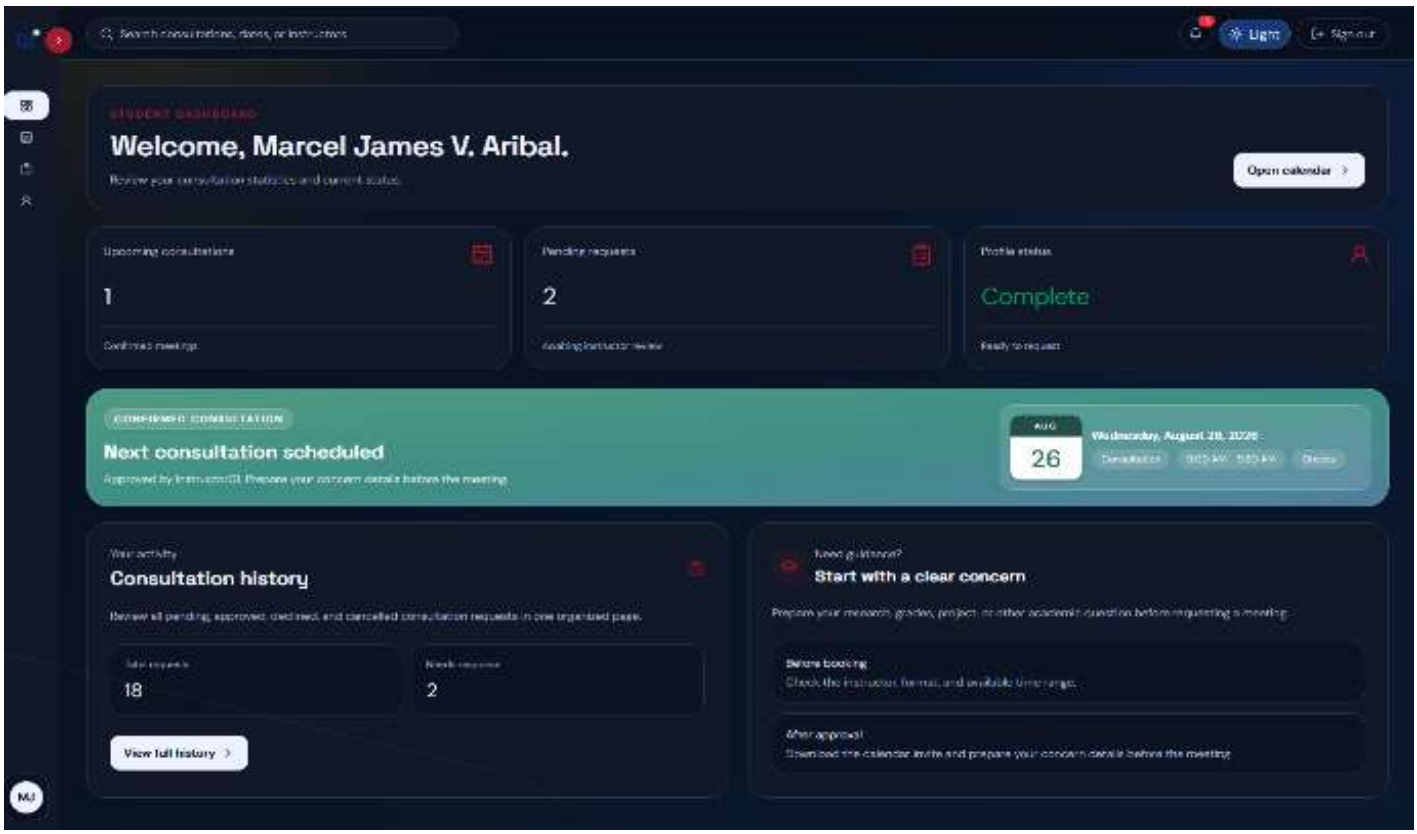
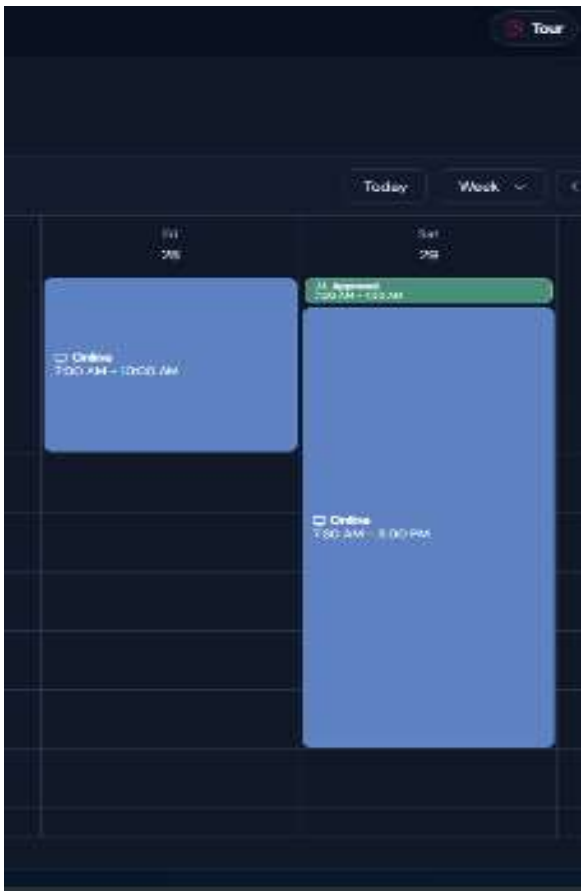
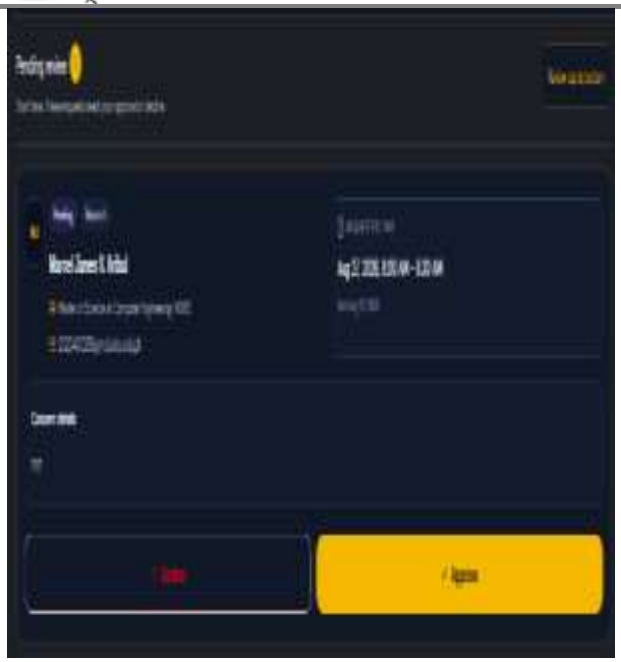


Figure 8 System's Main Features



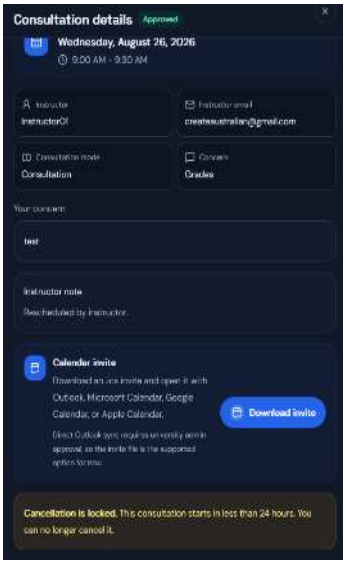


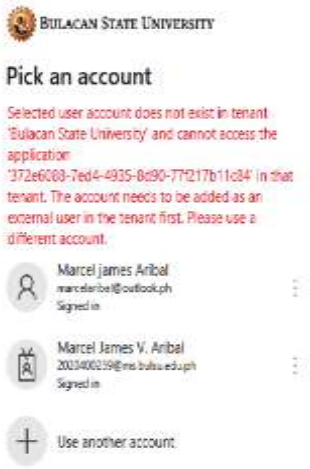
A. 1-Day Cancellation & Rescheduling Engine B. Schedule Conflict Locking Engine

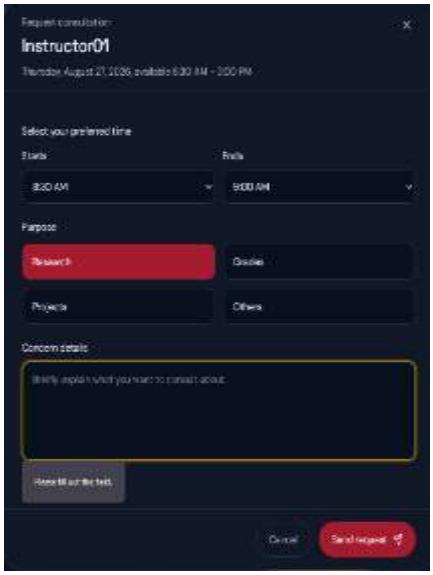


C. Approving/Declining Feature

Figure 9 System's Input Validation, Error Handling, and Confirmation Messages







A. Confirmation message B. Unauthorized login C. Missing required field alert

Table 3 Requirement Compliance Matrix

Feature / requirement	User, input, and process	Output	Status/evidence
FR-01	User signs in using Microsoft authentication	An authenticated session started	MT-001
FR-02	System checks login and looks for an existing user profile	A user profile was created (new user) or retrieved (returning user)	MT-001
FR-03	Students fill out required profile fields	The completed profile was saved	MT-003; ST-004

FR-04	Instructor sets up consultation availability windows	A schedule was posted	MT-002
FR-05	Instructor selects consultation mode (F2F, Online, or Both)	The mode setting was saved	MT-002
FR-06	Instructor limits availability to specific program(s)	The program-restricted availability was saved	MT-002
FR-07	System matches student's program to instructor availability	Only the relevant availability slots were shown to student	MT-002
FR-08	Student selects an open time window and submits a request	The consultation request was created	MT-003; ST-001
FR-09	Student enters concern type and concern details (required fields)	Request save along with concern information	MT-003; ST-004
FR-10	Instructor reviews and approves or declines a request	Request status updated	MT-004; ST-002
FR-11	System checks new approval against existing approved schedules	Overlapping approvals blocked/prevented	MT-005; ST-002, ST-006; ST-007
FR-12	System triggers a notification on approval, decline, or cancellation	A notification was sent to student	MT-004; ST-015
FR-13	System triggers a notification when a request is submitted	A notification was sent to instructor	MT-004; ST-001
FR-14	User searches consultation history	List of matching past consultations displayed	ST-002; ST-011
FR-15	User downloads calendar invite	The downloadable calendar file was provided to the user	ST-016
FR-16	User attempts to cancel a consultation before the cutoff period	Cancellation processed (blocked after cutoff)	MT-006; ST-008
FR-17	System processes new data (requests, approvals, etc.)	The dashboard refreshed in near real time	ST-001; ST-002; ST-011
NFR-01	User accesses system from desktop, tablet, or mobile	Responsive layout displayed correctly	ST-013
NFR-02	System checks user role before granting access	Access allowed/denied based on role	MT-001; ST-005; ST-014
NFR-03	System applies Supabase Row Level Security to database queries	Unauthorized record access blocked	ST-001; ST-002
NFR-04	System validates booking against existing approved bookings	Duplicate/conflicting bookings prevented	MT-005; ST-002, ST-006; ST-007
NFR-05	System logs and stores consultation records	Historical records maintained for accountability	ST-002; ST-011
NFR-06	System verifies accounts against institutional accounts	Access restricted to verified institutional accounts	MT-001; ST-001

The compliance matrix showed that all 17 functional requirements and all 6 non-functional requirements were fully implemented.

Table 4 Test Cases Used for System Verification

Test ID	Requirement/ scenario	Expected result	Actual result	Initial	Action/retest
MT-001	User attempts to login using a personal email account.	Access rejected with error: "Access restricted to valid @ms.bulsu.edu.ph accounts."	Users were incorrectly granted access without domain filtering.	Failed	Added backend domain validation logic in Supabase auth callback; retested successfully
MT-002	Instructors set availability strictly for "Graduate School"	Non- Graduate School students are blocked from selecting the slot; Graduate School students can proceed.	System filtered access accurately based on the user's student profile.	Passed	None; module functioning as expected.

	program students.				
MT-003	Student A attempts to book a consultation with complete details.	Booking will be queued in the pending list, if all required fields are specified.	System labeled the request "Pending" as it waits for the instructor's decision.	Passed	None; module functioning as expected.
MT-004	Instructor A attempts to approve a pending booking request.	Booking shall proceed and users will be notified about it and lock the timeslot.	Slot locked and all involved users on the schedule are notified real-time.	Passed	None; module functioning as expected.
MT-005	Student B attempts to book a timeslot that Student A was just approved for.	Approved slot locks automatically and prevent Student B from booking.	Slot disabled in real-time; booking submission blocked.	Passed	None; module functioning as expected.
MT-006	Student attempts to cancel an approved appointment scheduled 5 hours away.	The system blocks cancellation and displays an error: "Cancellations must be made at least 1 day in advance."	The system blocked action and showed a warning prompt.	Passed	None; module functioning as expected.
ST-001	A student login, completes profile, browses calendar, and requests consultation on chosen schedule	The flow from login to booking should have no error and will be tagged as "Pending"	Worked as expected; booking completed in under 10 seconds.	Passed	None; functioning as expected.
ST-002	Instructor approves a request submitted by a student in a different device/browser /session	Approved status and locked slot should reflect in real-time on both dashboards	Worked as expected; status correctly updated within 1-2 seconds	Passed	None; functioning as expected.
ST-003	User navigates across the dashboard, calendar, and other pages	Pages should load with consistent layout and no broken links	Worked as expected; consistent layout upon navigation	Passed	None; functioning as expected.
ST-004	Student submits a consultation request without indicating a concern	System should block submission and display an inline error message	Request blocked; error message displayed to supply missing details	Passed	None; functioning as expected.
ST-005	An already graduated	The system should not allow the login as they are	System allowed the login; graduated	Failed	See BUG-01

	BulSU student tried logging in and booking an appointment	not in the program anymore.	user was able to request a consultation		
ST-006	Student A and B booked an exact same timeslot	System accepts the bookings but only the approved request will be plotted on the calendar	Worked as expected; system accepted the booking while locking only the approved request	Passed	None; functioning as expected.
ST-007	Student A and B booked an overlapping timeslot on same date	System accepts the bookings but only the approved request will be plotted on the calendar	Worked as expected; system accepted the booking while locking only the approved request	Passed	None; functioning as expected.
ST-008	An attempt to cancel an approved appointment scheduled 5 hours away	System should block the schedule cancellation and display an error message	Worked as expected; system blocked action and showed warning prompt	Passed	None; functioning as expected.
ST-009	Student attempts to make a booking for a past date on the calendar	System should not allow the booking as its already past the calendar date	System allows the booking even if it is past the calendar date	Failed	See BUG-02
ST-010	Instructor attempts to remove some of their plotted available time including a confirmed booking consultation	System should allow the editing or removal of some plotted schedule, but a confirmed booking should remain	Worked as expected; system allowed the editing of plotted time while making sure that confirmed booking will remain	Passed	None; functioning as expected.
ST-011	Two instructor sessions are opened simultaneously , viewing the same pending request queue	Both sessions should reflect the same, up-to-date queue without a manual refresh	Worked as expected; system maintained up-to-date information for both without lag	Passed	None; functioning as expected.
ST-012	Student's session token expires mid-booking and the student attempts to submit again	System should redirect to sign-in and preserve the unsaved form data for resumption	User was redirected to sign-in, but unsaved form data was lost	Failed	See BUG-03
ST-013	User accesses the system's dashboard and booking function using a mobile-sized viewport	System layout should be responsive with all buttons and forms usable and still intact	Worked as expected; system shows responsiveness with different viewports	Passed	None; functioning as expected.

ST-014	User attempts to login using a personal email account	System should not accept unauthorized users and display error message	Worked as expected; system blocked the login of an unauthorized domain	Passed	None; functioning as expected.
ST-015	Student's booking request was already approved by the instructor	System should send an automated email for booking confirmation	System sent an in-system alert but no automated email confirmation	Failed	See BUG-04
ST-016	Successful booking was made	System should allow users to add the meeting details on their calendar, and an automatic meet link will be generated for online consultations	System allows manual calendar invite integration but does not create an automatic meeting link for online consultations	Failed	See BUG-05

Of the twenty-two (22) final system test cases, sixteen (16) passed during the initial run and six (6) failed, producing an initial pass rate of 72.73%. The failed cases involved graduated BulSU students with valid BulSU email addresses may still login on to the system, booking past a calendar date, error handling on expired sessions, automated email notification for updates, and automatic generation of meeting link for online consultations.

Table 5 Bugs and Fixes Applied

Bug	Description/module	Severity/cause	Correction	Retest/status
BUG-01	Graduated Student with Valid BulSU Email Account Login	Minor	Microsoft Entra cannot auto-detect current students and graduated	Unresolved
BUG-02	Booking a Past Calendar Date	Minor	Selectable date ranges were limited only to future dates	Resolved
BUG-03	Error Handling on Expired Sessions	Minor	Session time is extended up to 1 hour	Unresolved
BUG-04	Email Notification for Booking Updates	Minor	Implemented the module for email automation	Resolved
BUG-05	Automatic Meeting Link Generation for Online Mode	Minor	Instead of auto-generation of meeting links, Instructors shall provide links (optional) upon plotting their available schedule.	Resolved

After limiting the selectable date range, implementing a module for the email automation, and adding manual meeting link input option, three (3) cases passed retesting and two (2) remained unresolved.

The system addressed the core problem of manual, informally coordinated consultation scheduling between graduate school students and professors through six fully integrated features (institutional single sign-on, instructor availability, program-scoped scheduling, student booking, an approve/decline workflow, and an automated schedule conflict locking engine) all verified through module, integration, and system-level testing and confirmed against the approved SRS in the Requirements Compliance Matrix, where all 17 functional requirements and all 6 non-functional requirements were marked fully implemented. However, two remaining limitations constrain the interpretation of these results: the system could not automatically distinguish between currently enrolled and graduated students during SSO authentication (BUG-01), and unsaved booking data was lost when a user's session expired mid-submission (BUG-03).

The findings are consistent with Ampuan and Deleña (2022) and Hashim et al. (2019) regarding the measurable convenience gains that digitized, role-based appointment systems offer over manual or walk-in coordination, since testing showed the booking-to-approval flow completing without error under expected coordination time (ST-001, ST-002); they are also consistent with Remigio (2024), who found that Philippine higher-education

institutions most often fall short in role-based access enforcement rather than in formal policy documentation, as the BulSU Consult Hub's Supabase Row-Level Security policies were specifically designed to close this gap and were confirmed functioning correctly during testing (ST-005, ST-014).

CONCLUSIONS AND RECOMMENDATIONS

Based on the completed development and recorded tests, the team designed and implemented BulSU Consult Huub with institutional single sign-on restricted to verified BulSU accounts, instructor availability and program-scoped schedule management, student consultation booking, an approve/decline workflow, an automated schedule-conflict locking engine, a one-day advance cancellation and rescheduling policy, and near real-time dashboard synchronization. The four phases provided a traceable progression from concept and requirements to module development, integration, and system verification. The prototype fully satisfied 15 out of 17 functional and all 6 non-functional requirements. Therefore, the evidence supports the conclusion that, under the conditions and scope tested, the BulSU Consult Hub can reliably enforce institution-restricted access, prevent double-booked consultation schedules, and maintain role-based data access between students and professors. It does not establish broader effectiveness, long-term reliability, or user acceptance beyond the actual testing scope.

Before full deployment, the group should resolve the inability to distinguish graduated from currently enrolled students during SSO authentication (BUG-01) and the loss of unsaved booking data when a session expires mid-submission (BUG-03), expand testing across wider range of devices and actual student and faculty participants, strengthen enrollment-status verification, improve accessibility for users relying on assistive technologies, and establish maintenance procedure for long-term reliability. Future developers may implement automatic calendar synchronization, native meeting link generation and messaging platform, actual mobile application, and limit the calendar scope based on the school calendar, conduct formal usability or acceptance testing with defined participants and instruments, and evaluate the system under real operating conditions.

ACKNOWLEDGMENT

The authors acknowledge Bulacan State University, the Graduate School, and the course instructor.

REFERENCES

1. Ampuan, A., & Deleña, R. (2022). An implementation and evaluation of web-based appointment system for the Mindanao State University – Main Campus. Journal of Information Systems and Informatics. https://doi.org/10.51519/journalisi.v4i4.379
2. Azhariyah, S., & Muhammad Mukhlis. (2024). Framework CSS: Tailwind CSS Untuk Front-End Website Store PT. XYZ. Jurnal Informatika, 3(1), 30-36. https://doi.org/10.57094/ji.v3i1.1601
3. Bass, L., Clements, P., & Kazman, R. (2021). Software architecture in practice (4th ed.). Addison-Wesley Professional. https://www.sei.cmu.edu/library/software-architecture-in-practice-fourth-edition
4. Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 1 September 1989; 13 (3): 319–340. https://doi.org/10.2307/249008
5. Fagarasan, C., Popa, O., Pisla, A., & Cristea, C. (2021). Agile, waterfall and iterative approach in information technology projects. IOP Conference Series: Materials Science and Engineering, 1169(1), Article 012025. https://doi.org/10.1088/1757-899X/1169/1/012025
6. Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps: Building and scaling high performing technology organizations. IT Revolution Press.
7. Fowler, M. (2018). Refactoring: Improving the design of existing code (2nd ed.). Addison-Wesley Professional. https://dl.acm.org/doi/10.5555/311424
8. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley Professional. https://dl.acm.org/doi/10.5555/186897
9. Hashim, M., Mailok, R., & Hassan, H. (2019). Pembangunan dan Pengujian Student Appointment and Consultation System (SAPCoS): Developing and Evaluting Student Appointment and Consultation System (SAPCoS). Journal of ICT in Education, 2, 90-100. https://ejournal.upsi.edu.my/JICTIE/article/view/2603

10. Humble, J., & Farley, D. (2010). Continuous delivery: reliable software releases through build, test, and deployment automation (pp. 7-92). Boston: Addison-Wesley. ISO/IEC/IEEE. (2017). ISO/IEC/IEEE 12207:2017: Systems and software engineering—Software life cycle processes. International Organization for Standardization. https://www.iso.org/standard/90219.html
11. Institute of Electrical and Electronics Engineers. (2024). IEEE standard for system, software, and hardware verification and validation (IEEE Std 1012-2024). IEEE.
12. ISO. (2019). ISO 9241-210:2019: Ergonomics of human-system interaction—Part 210: Human-centred design for interactive systems. International Organization for Standardization. https://www.iso.org/standard/77520.html
13. Jartarghar, H. A., Rao Salanke, G., A.R, A. K., G.S, S., & Dalali, S. (2022). React apps with server-side rendering: Next.js. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 14(4), 25–29. https://doi.org/10.54554/jtec.2022.14.04.005
14. Kravchenko, T., Bogdanova, T., Shevgunov, T. (2022). Ranking Requirements Using MoSCoW Methodology in Practice. In: Silhavy, R. (eds) Cybernetics Perspectives in Systems. CSOC 2022. Lecture Notes in Networks and Systems, vol 503. Springer, Cham. https://doi.org/10.1007/978-3-031-09073-8_18
15. Martin, R. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. https://www.researchgate.net/publication/234793613_Clean_Code_A_Handbook_of_Agile_Software_Craftsmanship
16. Miranda, E. (2023). MoSCoW rules. PM World Journal, 12(3). https://pmworldjournal.com/article/moscow-rules
17. Morris, L., & McDermott, L. (2020). Enhancing academic skills appointments through a new booking system. Journal of Learning Development in Higher Education, 19. https://doi.org/10.47408/jldhe.vi19.582
18. National Privacy Commission. (2012). Data Privacy Act of 2012 (Republic Act No. 10173). https://privacy.gov.ph/data-privacy-act/
19. Netguru. (2025). Supabase vs. Firebase: Which BaaS is best for your app? https://www.netguru.com/blog/supabase-vs-firebase
20. Nielsen, J. (1994). Usability engineering. Morgan Kaufmann. https://doi.org/10.1016/C2009-0-21512-1
21. Remigio, J. (2024). Implementation of Data Privacy Act of 2012 (RA 10173) in College of Criminal Justice Education of the University of Cagayan Valley. Aloysian Interdisciplinary Journal of Social Sciences, Education, and Allied Fields, 1(1). https://journals.aloysianpublications.com/index.php/articles/article/view/358
22. Saarikallio, M., & Tyrväinen, P. (2023). Quality culture boosts agile transformation—Action research in a business-to-business software business. Journal of Software: Evolution and Process, 35(1), Article e2504. https://doi.org/10.1002/smr.2504
23. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). Designing the user interface: Strategies for effective human-computer interaction (6th ed.). https://doi.org/10.1145/25065.950626
24. Zhao, P., Yoo, I., Lavoie, J., Lavoie, B. J., & Simoes, E. (2017). Web-based medical appointment systems: A systematic review. Journal of Medical Internet Research, 19(4), e134. https://doi.org/10.2196/jmir.6747

About The Authors

Marcel James V. Aribal is a graduate student in Master of Science in Computer Engineering at Bulacan State University. The author has skills in web development, programming, database management, user interface design and system testing. The author worked as the web developer for the project. They handled the process from design to development, implementation, testing and maintenance of the website. This included both end and back-end features. The author is interested in research and professional work related to web development, software engineering, information systems and new technologies.

Lorenz L. Banderado is a graduate student in the Master of Engineering Program in Computer Engineering at Bulacan State University. The author has skill in Circuit Assembly, Digital Logic Design, Programming

Languages and project responsibilities including leading hardware-software integration, Capstone Thesis Adviser. The author is interested in embedded systems design, IoT architecture, advanced computer hardware engineering and teaching young professionals.

Karen Kei A. Zapanta is a graduate student in Master of Science in Computer Engineering at Bulacan State University. The author has skills in UI/UX design, technical documentation, and instructional design, as well as a foundational background in electronics, including circuit analysis, digital logic design, and microcontroller-based systems. Project responsibilities included preparing the group's requirements, testing, and progress documentation. Research and professional interests include software design, educational technology, and the application of software engineering practices in developing institution-scale academic systems.

Dr. Ma. Magdalena V. Gatdula is currently the University Registrar of Bulacan State University. She earned her Bachelor of Science in Computer Engineering degree from the Polytechnic University of the Philippines. She completed her Master of Arts in Education major in Computer-Aided Instruction and Programming, Master of Science in Computer Engineering, and Doctor of Technology. She also served as Dean of the College of Engineering and the College of Information and Communications Technology at Bulacan State University.