Test Plan

Simon Garcia
Javier Paz
Joshua Sayur
Maria José Betancourt
Catalina Bernal

Table of contents

Introduction

This document presents the test plan for the project assigned in the context of Proyecto Integrador I at Universidad Icesi. The project consists of the design and implementation of a comprehensive web platform for Bienestar Universitario (BU), whose main purpose is to centralize, optimize, and modernize the processes currently managed manually by the university's BU office.

This application seeks to provide a single, unified environment where students, professors of the various activities offered by Bienestar Universitario, any other professors of the university, and administrators can interact efficiently with the different services offered by BU. Through the platform, users will be able to access cultural, human development, and sports activities, register for tournaments, participate in volunteer programs, and schedule BU services such as psychological counseling sessions. Additionally, the system integrates functionalities that allow students to manage their personal schedules, receive personalized notifications according to their interests, and stay informed of upcoming events.

Beyond the student perspective, the platform also includes tools for administrators and coordinators, such as the Analytics and Reports module, which consolidates data from different sources and generates performance indicators. These features will facilitate decision making processes by providing insights into participation rates, attendance levels, and general trends within the university community. By replacing external tools with an internally developed solution, the application strengthens the autonomy and adaptability of Universidad Icesi’s technological ecosystem.

Ultimately, the context of this project highlights the need for a vigorous and user-friendly digital system that not only improves the quality of communication and coordination within BU but also ensures long term sustainability, scalability, and student engagement.

Test objectives

The primary objective of conducting tests throughout the development of the BU platform is to ensure the reliability, quality, and overall effectiveness of the system before its release to the university community. Testing will validate both functional and non-functional requirements, ensuring that all features operate as intended and that the platform provides a secure and seamless user experience.

Specifically, the testing objectives are:

1. Validation of functional requirements: Ensure that modules such as activity registration, tournament management, appointment scheduling, notifications, and reporting behave according to the specifications and user stories defined during the requirements analysis phase.

2. Verification of system integration: Confirm that all subsystems interact correctly, exchanging data smoothly and maintaining consistency across different modules of the platform.

3. Usability and accessibility assurance: Evaluate that the interface is intuitive, user-friendly, and accessible to the entire university community, taking into account students, professors, and administrative staff.

4. Performance and scalability testing: Verify that the application can handle the expected number of concurrent users without degradation of performance, and that it can be scaled as adoption grows.

5. Security and data protection: Validate that personal and institutional data are properly protected through authentication mechanisms, secure storage, and compliance with relevant privacy standards.

6. Cross-environment compatibility: Ensure that the system operates correctly across different devices (desktop, tablets, smartphones) and browsers commonly used by the student body.

7. Early detection and correction of defects: Identify inconsistencies, bugs, or vulnerabilities during development to minimize risks, reduce costs, and ensure a smooth deployment process.

In conclusion, the testing process is not only a mechanism to detect and correct failures, but also a strategic activity that supports the achievement of the project's main goal: delivering a reliable, secure, and high-quality platform that meets the expectations of the Universidad Icesi community and enhances the services provided by Bienestar Universitario.

Testing strategy for the project

Work Scheme

1. Test Planning

    1.1. Review of Requirements and User Stories: We will begin by reviewing the requirements and user stories defined for the BU platform in order to fully understand the functionalities that need to be tested. This review will take place during sprint planning meetings to ensure that the entire team is aligned with the testing objectives. The requirements include critical functionalities such as activity registration, tournament management, psychological appointment scheduling, personalized notifications, and data analysis through reports. Following the order of most critical functions which need to be developed first and the least critical functions which can to be developed at last.

    1.2. Definition of Acceptance Criteria: For each user story, we will establish clear acceptance criteria that define the standards each functionality must meet to be considered complete. For example, acceptance criteria for tournament registration will ensure that users can successfully register individually or as a team, that eligibility is verified, and that schedules and results are updated in real time.

    1.3. Use of Gherkin for Test Scenarios:We will use the Gherkin language to define test scenarios in a readable and collaborative way. This will allow both technical and non-technical stakeholders, such as BU administrators, to understand and validate test cases. For instance, scenarios like "Given a student wants to schedule a psychological appointment, when they check the availability, then they should see all available time slots in real time" will directly connect the requirements with the test execution.

2. Test-Driven Development (TDD)

    2.1. Writing Initial Test Cases: Before coding begins, initial test cases will be written based on the acceptance criteria. For example, test cases will validate that students can add and remove activities from their personal calendars, that notifications are triggered according to preferences, and that reports correctly consolidate participation data.

    2.2. Iterative Development and Testing: Functionalities will be implemented in short cycles, where each cycle includes writing the necessary code and executing the corresponding tests to verify proper behavior. For instance, when developing the notification system, we will first test the logic for sending reminders, then expand to cover personalization, and finally validate its integration with the academic calendar. This iterative approach will allow early detection of issues and immediate adjustments.

3. Testing Process

    3.1. Unit Testing Execution: Unit tests will be performed to verify that individual functions, methods, and classes such as the module for calculating tournament standings or the function for generating attendance records work correctly in isolation. This step ensures that small components are error free before integration.

    3.2. Integration Testing Execution: Once unit tests are successful, integration tests will confirm that the different modules of the platform like the activity registration system, the scheduling system, and the reporting engine work together cohesively and fully integrate with one another. For example, registering in a tournament should automatically update the student's personal calendar and generate confirmation notifications.

    3.3. Functional Testing Execution: With integration confirmed, functional tests will validate that the system as a whole meets the acceptance criteria of the user stories. This includes confirming that students can complete full workflows, such as registering for a tournament, receiving notifications, and later accessing participation statistics.

    3.4. Security Testing Execution:Security tests will be carried out to identify vulnerabilities, ensure proper handling of sensitive information (such as personal data in psychological appointments), and confirm that authentication and authorization mechanisms work as expected.

3.5. Usability Testing Execution: Finally, usability tests will be conducted with representative end-users (students, professors, and administrators) to evaluate the ease of use of the interface, the clarity of workflows, and the overall user experience. Feedback will be collected to make improvements that ensure the platform is intuitive and engaging.

4. Constant Communication

4.1. Effective Communication: We will maintain constant and effective communication within the team regarding the progress of tests, the issues identified, and the solutions applied. Transparency in communication will be key to fostering collaboration among developers, testers, and BU stakeholders, ensuring that problems are resolved quickly.

4.2. Regular Updates: All team members will be kept updated on the testing status through regular reports, daily stand-ups, and sprint review discussions. These updates will guarantee that the entire team remains informed and aligned with the project's objectives, while also providing visibility to BU administrators at Universidad Icesi regarding the quality and reliability of the platform being developed.

5. Test matrix

Type of test	Usability	Reliability	Security	Functionality
Unity tests		X: Verify that methods like scheduling, registration, and result recording work consistently.		X: Validate small components such as creating a new activity or saving a notification.
Integration tests		X: Ensure data consistency between modules (registering for a tournament updates the	X: Confirm secure data flow between login, scheduling, and reporting modules.	X: Verify workflows that combine multiple modules, like registering for an event and

		student's calendar).		receiving confirmation.
Functional tests	X: Validate that students can intuitively navigate the calendar and activities.	X: Ensure correct handling of edge cases (full tournaments or canceled activities).		X: Confirm that complete features (tournament management, scheduling, notifications) work as expected.
Security tests		X: Ensure session management under heavy loads.	X: Verify authentication, role-based access (student, professor, admin), and data protection for psychological appointments.	X: Confirm system functions are not compromised when security mechanisms are applied.
Usability tests	X: Evaluate ease of use of the interface with real students and administrators.			X: Validate that essential tasks (registering for events, checking schedules, receiving notifications) can be completed efficiently.

Support Tools

1. Pytest

    1.1. Description: A widely used Python testing framework that is simple to learn and provides clear, readable test cases. In the BU platform, Pytest will be used to validate core functionalities such as activity registration, notification delivery, and appointment scheduling.

1.2. Suitable for: Unit and functional tests in Django modules.

1.3. Documentation: Pytest Documentation

2. Unittest (Python Standard Library)

2.1. Description: Python's built-in testing framework, similar to JUnit in Java, which requires no additional installation. In the BU platform, it can be used by students to practice writing unit tests for small modules like calendar functions or report generators.

2.2. Suitable for: Unit testing without external dependencies.

2.3. Documentation: Unittest Documentation

3. Postman

3.1. Description: A graphical tool that allows testing and automating API calls with little to no code. In the BU platform, Postman will be used to verify that backend services (e.g., tournament registration, psychological appointment scheduling, analytics endpoints) respond correctly and handle errors gracefully.

3.2. Suitable for: Integration and security testing of BU's APIs.

3.3. Documentation: Postman Learning Center

4. Trello / Jira (Student Version)

4.1. Description: Free task management platforms for organizing and monitoring projects. In the BU platform, Trello or the free Jira version will be used to track test execution, document defects, and follow up on testing progress across sprints.

4.2. Suitable for: Managing test tasks, defect tracking, and progress reporting.

4.3. Documentation: Trello Documentation / Jira Software Guide

5. Google Lighthouse

5.1. Description: An auditing tool integrated into Google Chrome to evaluate

performance, accessibility, and usability of web applications. In the BU platform, Lighthouse will be used to analyze usability and user experience for modules such as activity registration, schedules, and student dashboards.

5.2. Suitable for: Usability and performance testing of the BU web interface.
5.3. Documentation: Lighthouse Documentation

6. SQLite + DB Browser

6.1. Description: A lightweight database system with a graphical interface for reviewing and testing data. In the BU platform, SQLite with DB Browser will be used to populate test data (students, activities, appointments, reports) and check data consistency without requiring a complex database setup.

6.2. Suitable for: Database testing and quick validation of stored test data.

6.3. Documentation: DB Browser Documentation

Types of Tests to Apply

1. Unit Tests: Each individual component of the BU system will be evaluated to confirm that it works correctly. This involves designing and executing specific test cases for each function, method, or class, verifying that they behave as expected. Examples include validating the creation of activities, registering a new user, or generating a notification.

1.1. Responsible: The development team, with each member in charge of creating and executing tests for the code units they have developed. The team leader will supervise and coordinate this process.

2. Integration Tests: The interaction between the different modules of the BU system will be evaluated to ensure their compatibility, proper communication and order flow. The goal is to guarantee that all components such as tournament registration, personal scheduling, notifications, and reporting work together effectively and that data is correctly transferred between them. For example, registering for a tournament should automatically update the student's personal calendar and trigger a confirmation notification.

2.1. Responsible: The development team will collaborate in the design and execution of tests to validate the integration of the modules. The team leader will supervise and coordinate these tests.

3. Functional Tests: The functionalities of the BU system will be checked and compared to the acceptance criteria of the user stories. This includes verifying that the software meets the expectations and needs of students, professors, and administrators. For instance, functional tests will confirm that students can book psychological appointments, administrators can view participation reports, and all notifications are sent at the right time.

    3.1. Responsible: The development team will design and execute tests that ensure compliance with the acceptance criteria. The team leader will coordinate this process and ensure that the established requirements are met.

4. Security Tests: The security of the BU platform will be evaluated, focusing on aspects such as user authentication, role based access (student, professor, admin), and the protection of sensitive information, including data from psychological sessions. Security tests will also aim to detect vulnerabilities and confirm that data encryption and secure communication are implemented.

    4.1. Responsible: The development team will detect and correct possible security vulnerabilities in the code. The team leader will supervise this process and ensure that the necessary security measures are applied.

5. Usability Tests: The ease of use and user experience of the BU platform interface will be evaluated. These tests will involve gathering feedback from real users (students, professors, and administrators) to identify areas of improvement in the interface and overall experience. For example, testers will validate whether the process of finding and registering for an activity is intuitive and efficient.

    5.1. Responsible: End users will conduct usability tests, providing feedback on the system's experience. The development team will use this feedback to improve the interface and user experience. The team leader will supervise this process and coordinate the implementation of suggested improvements.

Test Plans Reach

All functionalities will be validated through two primary types of tests:

1. Unit Tests: To verify that each individual component of the software works as expected in isolation.
2. Integration Tests: To ensure that all modules of the application work together correctly as a unified system.

Epic	HU	Functionality	Quality Attribute	Types of tests
Activity and Event management (CADI)	1.1	Publication of cultural/sports activities (without registration)	Functionality, Usability, Reliability	Unit, Integration
	1.2	Viewing activities by filters (type, place, time)	Functionality, Usability, Reliability	Unit, Integration
	1.3	Interactive calendar of activities	Functionality, Usability, Reliability	Unit, Integration
	1.4	Feedback collection (ratings, comments)	Functionality, Usability, Reliability	Unit, Integration
Events & Services	2.1	Tournament registration (individual/team)	Functionality, Usability, Reliability, Security	Unit, Integration
	2.2	Tournament results and rankings	Functionality, Usability, Reliability	Unit, Integration
	2.3	Registration in volunteer/social projects	Functionality, Usability, Reliability	Unit, Integration
	2.4	Monitoring participation in projects	Functionality, Usability, Reliability	Unit, Integration
	2.5	Chatbot support for PSU queries	Functionality, Usability	Unit, Integration
	2.6	Scheduling/canceling psychological appointments	Functionality, Usability, Reliability, Security	Unit, Integration
Information and	3.1	Personal	Functionality,	Unit, Integration

notifications		schedule management (add, edit, delete activities)	Usability, Reliability	
	3.2	Unified calendar of all BU activities	Functionality, Usability, Reliability	Unit, Integration
	3.3	Sending personalized notifications (interests, reminders)	Functionality, Usability, Reliability, Security	Unit, Integration
Analytics and reports	4.1	Integration of internal/external data (CSV, Excel, DB)	Functionality, Reliability, Security	Unit, Integration
	4.2	Student participation analytics (frequency, type)	Functionality, Reliability	Unit, Integration
	4.3	Comparative metrics (by semester, faculty, gender)	Functionality, Reliability	Unit, Integration
	4.4	Export dashboards/reports (CSV, Excel)	Functionality, Reliability	Unit, Integration
	4.5	Attendance management (QR, manual, ID)	Functionality, Usability, Reliability, Security	Unit, Integration
	4.6	Automated alerts/communications	Functionality, Usability, Reliability, Security	Unit, Integration

Estimated Effort

Estimación de esfuerzo		**Complejidad**		
HU	**Funcionalidad**	**Baja**	**Media**	**Alta**
1.1	Publication of cultural/sports activities	1h	2h	3h
1.2	Viewing activities with filters (type, place, time)	1h	2h	3h
1.3	Interactive calendar of activities	1h	2h	3h
1.4	Collecting feedback (ratings, comments)	1h	2h	3h
2.1	Tournament registration (individual/team)	1h	2h	3h
2.2	Tournament results and rankings	1h	2h	3h
2.3	Registration in volunteer/social projects	1h	2h	3h
2.4	Monitoring participation in projects	1h	2h	3h
2.5	Chatbot support for PSU queries	1h	2h	3h

2.6	Scheduling/canceling psychological appointments	1h	2h	3h
3.1	Personal schedule management (add, edit, delete)	1h	2h	3h
3.2	Unified calendar of all BU activities	1h	2h	3h
3.3	Sending personalized notifications	1h	2h	3h
4.1	Integration of internal/external data (CSV, Excel, DB)	1h	2h	3h
4.2	Student participation analytics (frequency, type)	1h	2h	3h
4.3	Comparative metrics (semester, faculty, gender)	1h	2h	3h
4.4	Export dashboards/reports (CSV, Excel)	1h	2h	3h
4.5	Attendance management (QR, manual, ID)	1h	2h	3h
4.6	Automated alerts/communications	1h	2h	3h

Process deliverables

1. Test plan

    1.1. Description: A document that defines the objectives, scope, strategy, and resources required for testing the Bienestar Universitario (BU) platform at Universidad Icesi. It includes details about the types of tests to be applied (unit, integration, functional, security, usability), the acceptance criteria for each module (tournament registration, appointment scheduling, analytics reports), and the roles and responsibilities of the development and testing team.

2. Test cases

    2.1. Description: Documents that specify the conditions, steps, required data, and expected results for each test. These cases will guide the execution of tests and ensure that the BU platform meets both functional and non-functional requirements. For example, a test case may verify that a student can successfully schedule a psychological appointment, or that registering for a cultural event updates their personal calendar correctly.

3. Test reports

    3.1. Description: Reports that summarize the results of the tests carried out on the BU platform, including the status of each test (passed, failed, pending), the problems encountered, and the severity of the defects. These reports provide an overall view of the system's quality after testing, allowing Universidad Icesi to assess whether the platform is ready for deployment.

4. Defect documentation

    4.1. Description: Detailed records of defects found during the testing of the BU platform, including a description of the defect, the steps to reproduce it, screenshots or evidence when applicable, and the current resolution status. This documentation facilitates defect management and ensures that problems such as incorrect tournament rankings, failed notifications, or errors in exporting reports are corrected efficiently.

5. Test Metrics

    5.1. Description: Quantitative data about the BU testing process, such as the number

of executed, failed, and passed tests, coverage percentage by module (activities, events, notifications, analytics), and the time invested in testing. These metrics will help evaluate the effectiveness of the testing process and support decision making regarding improvements in the platform's quality assurance.

Follow up and control mechanisms

1. Follow up meetings

    1.1. Frequency: Weekly

    1.2. Participants: All members of the BU development and testing team (developers, testers, and project leader).

    1.3. Objective: Review the progress of tests for modules such as activity registration, psychological appointment scheduling, and analytics reports; identify issues (failed notifications, integration errors) and define corrective actions.

    1.4. Format: Presentation of test progress, key metrics (coverage, defects found), and discussion of obstacles with proposed solutions.

2. Completeness Indicators

    2.1. Description: Indicators used to evaluate the status of BU testing against established objectives. Examples include: percentage of functional coverage (number of features validated in CADI, events, or notifications), number of critical vs. minor defects identified, and usability success rates in student tests.

    2.2. Evaluation Frequency: Continuous, with formal reviews during weekly meetings.

3. Test Status Reports

    3.1. Frequency: Weekly.

    3.2. Content:
        3.2.1. Code coverage by module (activities, services, analytics, notifications).
        3.2.2. Percentage of successful integrations (event registration updates calendar correctly).
        3.2.3. Percentage of test cases passed vs. failed.
        3.2.4. Severity level of vulnerabilities (in authentication, data privacy for

psychological appointments).

3.2.5. Average time required to complete specific testing tasks.

3.3 Purpose: Provide Universidad Icesi and the project team with updated information to make informed decisions, prioritize defect resolution, and guarantee the quality of the BU system.

4. Updates in Jira (or equivalent tool)

4.1. Frequency: Daily.

4.2. Content: Registration of defects detected during BU testing (incorrect tournament rankings, errors in exporting reports, or missing notifications), completed tasks, and progress in test case execution.

4.3. Purpose: Maintain an up to date record of the project's testing status, improve traceability, and facilitate coordination among team members during the BU platform development.

Specific Metrics per Test Type

Test type	Metric	Description	Formula	Tool	Frequency
Unit tests	Code coverage	Measures the proportion of BU platform code executed during unit tests (appointment scheduling, activity registration).	$\frac{Covered\ code\ lines}{Total\ code\ lines}$	Pytest/Unittest + Coverage plugin	Weekly
Integration tests	Successful integration percentage	Measures the proportion of integrations between BU modules (registering for a tournament	$\frac{Successful\ integrations}{Total\ integrations}$	Postman (for API calls)	Bi-weekly

		updates the student calendar) that succeed without errors.			
Functional tests	Percentage of passed test cases	Measures the proportion of functional test cases executed successfully (notification delivery, event filtering, exporting reports).	$\frac{Passed\ test\ cases}{Total\ vulnerabilities\ found}$	Pytest (Django tests)	Weekly
Security tests	Severity of vulnerabilities found	Evaluates the risk of vulnerabilities (weak authentication, insecure data in appointments) by assigning severity scores.	$\frac{Sum\ of\ severity\ scores}{Total\ vulnerabilities\ found}$	Postman (auth & headers) + manual checks	Monthly (as deemed necessary)
Usability tests	Average time to complete a task	Measures how long, on average, a user takes to complete tasks in BU (booking an appointment, registering for activities).	$\frac{Total\ task\ time}{Number\ of\ tasts}$	Google Lighthouse (UX audit)	Every two weeks
Database tests	Data consistency checks	Ensures that test data (students, events, appointments) is correctly stored and retrieved during testing.	$\frac{Consistent\ records}{Total\ records\ checked}$	SQLite + DB Browser	On demand (per sprint or as deemed necessary)

Test Cases

Login application:

Scenario configurations:

Name	Class	Scenario

setup	login/views	• Empty Client()
		• Faculty = "Systems Engineering"
		• Faculty_bool = False
		• Role_basic = "basic user"
		• Basic_role_bool = False
		• User:
		  ○ Username = "testuser"
		  ○ Password = "test123"
		  ○ Email = "test@example.com"
		  ○ Faculty = Faculty
		  ○ Role = role_basic

Test Cases Design

Test's objective: Object creation from initial app setup				
Class	Method	Scenario	Input	Expected Output
login/apps.py	create_initial_data()	setup	N/A	get_or_create() for faculty and role_basic should return false since they were created by apps.py

Test's objective: Validate the login page				
Class	Method	Scenario	Input	Expected Output
login/views.py	login_view()	setup	Search for "login:login" with the client	Status code = 200 and it should contain the words

				“Bienvenido de nuevo” from the login page
login/views.py	login_view()	setup	Username = “testuser” Password = “test123”	The login_views() should redirect the client towards the homepageUser since the basic user does exist.
login/views.py	login_view()	setup	Username = “wrong” Password = “wrongpass”	Status code = 200 even though the user doesn’t exist. The message “Please enter a correct username and password.” should appear too.
login/views.py	login_view()	setup	Username = “adminuser” Password = “adminpass”	The client should be redirected towards the homepageCADI made for admins

Test’s objective: Validate the registration page				
Class	Method	Scenario	Input	Expected Output
login/views.py	register_view()	setup	Search for “login:registration” with the client	Status code = 200 and it should contain the words “Crear cuenta” from the registration page
login/views.py	register_view()	setup	New user { Username = “newuser”	The client should be redirected

			Firstname = “new” Lastname = “user” Email = “new@example.com” faculty= Id of Systems Engineering Password1 = “strongpassword123” password2= “strongpassword123” }	towards login:login and there should be a new User Object with username “newuser”

login/views.py	register_view()	setup	New user {
			Username = “baduser”
			Firstname = “bad”
			Lastname = “user”
			Email = “bad@example.com”
			faculty= Id of Systems Engineering
			Password1 = “pass1234”
			password2= “pass5678”
			
			}
				Even though the password validation is wrong, a status code = 200 should be returned, the message “The two password fields didn’t match.” should be returned and there shouldn't be a new object in the DB with username “baduser”

Activities application:

Scenario Configurations

Name	Class	Scenario	Input / Setup
Empty Activity	Activity	Activity with only name “Fútbol”	Activity(name="Fútbol")
User	User	Username = “tester1”, Password = “12345”	User.objects.create_user(...)
Enrolled Activity	Activity	Activity “Yoga” with max_capacity=1	Activity.objects.create(name="Yoga", max_capacity=1)
Published Activity	Activity	Activity “Pintura”	Activity.objects.create(name="Pintura", is_published=True)
Client with no authentication	Client	Anonymous user	self.client without login

Unit Tests

Test’s objective	Class	Method	Scenario / Setup	Input	Expected Output
Object creation from initial activity setup	ActivityModelTest	test_activity_str	Activity with name “Fútbol”	Activity(name="Fútbol")	String representation should be "Fútbol"
Validate full activity status	ActivityModelTest	test_activity_is_full_status	Activity with max_capacity=1 and one Enrollment linked to a user	Enrollment.objects.create(user, activity)	Should return True because the activity is full
Validate unique enrollment	EnrollmentModelTest	test_enrollment_unique_constraint	Duplicate enrollment for same user/activity	Enrollment.objects.create(user, activity) twice	Should raise IntegrityError
Validate schedule string	ScheduleModelTest	test_schedule_str	Schedule linked to activity “Pintura”	day=WeekDay.MONDAY, start_time=09:00, end_time=10:00	String should be "Pintura - Lunes (09:00:00 - 10:00:00)"
Validate review creation	ActivityReviewModelTest	test_activity_review_creation	Activity “Pintura”, user “tester1”	ActivityReview.objects.create(...)	String should be "Reseña de tester1 para Pintura (5⭐)" and is_read=False

Functional / Integration Tests

Test's objective	Class	Method	Scenario / Setup	Input	Expected Output
Validate activity list page	Activities TestSuite	test_activity_list_page	Client requests activity list page	GET /activities/	Status 200 and contains "Pintura"
Create activity via view	Activities TestSuite	test_create_activity_view	Authenticated user submits form	POST /activities/create/ with name, description, type, category, etc.	Status 302 redirect and Activity with name "Ajedrez" exists
Update activity via view	Activities TestSuite	test_update_activity_view	Authenticated user updates existing activity	POST /activities/update/<id>/ with new data	Status 302 redirect and activity name updated to "Pintura avanzada"
Delete activity via view	Activities TestSuite	test_delete_activity_view	Authenticated user deletes activity	GET /activities/delete/<id>/	Status 302 redirect and activity no longer exists
Enroll in activity via view	Activities TestSuite	test_enroll_in_activity_view	Authenticated user enrolls in activity	GET /activities/enroll/<id>/	Status 302 redirect and new Enrollment linking user and activity exists

Tournament Application:

Unit Tests

Test's objective	Class	Method	Scenario / Setup	Input	Expected Output
Create new tournament	TournamentModelTest	test_create_tournament	Tournament with initial parameters	Tournament.objects.create(name="Torneo Futbol", sport="Fútbol", gender="M", modality="E", start_date=today, max_participants=10)	Tournament created successfully, current_participants = 0, progress() = 0

Register individual participant	ParticipantTest	test_register_individual_participant	Tournament “Torneo A” (Tenis, modalidad I) with max 5 participants	Participant.objects.create(name="Ana Perez", tournament=tournament)	Participant added, current_participants = 1, progress() = 20.0
Edit participant name	ParticipantTest	test_edit_participant	Participant “Carlos” created in tournament	Update participant name to “Carlos M.”	Participant updated, name = "Carlos M."
Delete participant	ParticipantTest	test_delete_participant	Participant “Laura” enrolled in tournament “Torneo A”	participant.delete()	Participant removed, current_participants = 0
Register new team	TeamTest	test_register_team	Tournament “Torneo B” (Baloncesto, modalidad E)	Team.objects.create(name="Equipo A", members="Juan, Pedro, Luis", tournament=tournament)	Team created, Team.objects.count()=1, members string includes “Pedro”, current_participants = 1

Functional / Integration Tests

Test's objective	Class	Method	Scenario / Setup	Input	Expected Output
Validate tournament list page	TournamentsViewTests	test_tournaments_menu_view	Authenticated client requests list of tournaments	GET /tournaments/menu/	Status 200 OK and HTML contains tournament names
Create tournament via view	TournamentsViewTests	test_create_tournament_view	Authenticated user submits tournament form	POST /tournaments/create/ with valid form data	Redirect (status 302) and tournament appears in database
Create schedule via view	ScheduleViewTests	test_create_schedule_view	User submits schedule form	POST /schedules/create/ with valid date/time	Redirect (status 302) and Schedule object

					created
Register individual participant via view	ParticipantsViewTests	test_register_participant_individual	Tournament modality “I”	POST /tournaments/<id>/register/ with participant form	Redirect 302 and Participant linked to tournament
Register team participant via view	ParticipantsViewTests	test_register_team_participant	Tournament modality “E”	POST /tournaments/<id>/register/ with team form	Redirect 302 and Team linked to tournament
Edit participant via view	ParticipantsViewTests	test_edit_participant_view	Participant exists in tournament	POST /participants/edit/<id>/	Redirect 302 and participant updated
Delete participant via view	ParticipantsViewTests	test_delete_participant_view	Participant exists	GET /participants/delete/<id>/	Redirect 302 and participant removed

Edit team via view	TeamsViewTests	test_edit_team_view	Team exists	POST /teams/edit/<id>/	Redirect 302 and team updated
Delete team via view	TeamsViewTests	test_delete_team_view	Team exists	GET /teams/delete/<id>/	Redirect 302 and team deleted
View tournament results	ResultsViewTests	test_results_menu_view	Schedules exist with ended games	GET /tournaments/results/	Status 200 and shows only finished games
View ranking	RankingViewTests	test_ranking_view_individual	Tournament modality “I” with participants	GET /tournaments/ranking/<id>/	Status 200 and participants ordered by points and goal difference
View calendar	CalendarViewTests	test_calendar_general_view	Month with existing games	GET /tournaments/calendar/?month=10&year=2025	Status 200 and HTML

					contains scheduled matches