Design and Implementation of a Smart Project Balloting System Using Weighted Expertise Matching and Capacity-Constrained Workload Management

Dennis Unduikeye Ashishie, Benson Anita

Department of Computer Science, University of Calabar, Calabar, Nigeria

DOI: https://doi.org/10.51584/IJRIAS.2026.11090148

Received: 03 October 2026; Accepted: 08 October 2026; Published: 22 October 2026

ABSTRACT

The allocation of final-year students to project supervisors and topics is an important academic process that influences the quality of student research, the effectiveness of supervision, and the equitable distribution of departmental workload. In many tertiary institutions, however, this allocation is still performed manually, relying on physical balloting, paper forms, spreadsheets, and the personal judgment of coordinators. Such approaches are time-consuming, difficult to audit, prone to bias and human error, and often produce weak alignment between students' research interests and supervisors' expertise, as well as uneven distribution of students among supervisors. This study presents the design and implementation of a Smart Project Balloting System (SPBS), a web-based decision-support platform that automates the project allocation process. The system provides role-based access for administrators, supervisors, and students; supports project-topic proposal and approval; captures supervisor expertise and capacity declarations; collects ranked student preferences; and executes a weighted matching process that combines a keyword-based expertise score with a rank-based preference score under hard supervision-capacity constraints, using a greedy ranking procedure with a capacity-aware fallback. Workload is managed by enforcing declared capacities and monitoring utilization; the procedure does not minimize workload imbalance. The system was developed following the Prototype Development Methodology and implemented with a React/TypeScript browser interface, a server-side application layer, and a MySQL relational database. The prototype passed seven documented functional test cases. Allocation quality was additionally examined in a simulation of 100 synthetic cohorts (124 students, 18 supervisors) against a random baseline and a first-come balloting baseline, with an exact assignment solution as a reference. Relative to first-come balloting, the proposed procedure raised mean expertise compatibility from 0.35 to 0.59 and reduced assignments with no expertise overlap from 38.9% to 12.2%, but lowered first-choice satisfaction from 71.4% to 52.5%; it attained 97.2% of the optimal total score, left no student unmatched, and violated no capacity limit, while its utilization dispersion was comparable to the baselines. Because the simulation uses synthetic data, the study establishes functional feasibility and characterizes the expertise-preference trade-off; it does not claim experimentally established superiority over manual allocation in practice, and evaluation on real institutional data is identified as the principal direction for future work.

Keywords: Smart Project Balloting; Student–Supervisor Matching; Weighted Expertise; Capacity-Constrained Workload Management; Decision Support System; Project Allocation

INTRODUCTION

1.1 Background

Final-year project supervision is a central component of higher education. The student–supervisor relationship shapes research direction, the quality of academic feedback, student motivation, and ultimately the success of the project. The allocation of students to supervisors should therefore be treated as a strategic academic decision rather than a routine clerical task [1]. In practice, however, many departments still conduct project balloting and supervisor assignment manually, through physical balloting exercises, paper forms, spreadsheets, notice boards, and informal communication. Studies have consistently shown that such manual processes are slow, inconsistent, difficult to audit, and vulnerable to bias, and that they scale poorly as student cohorts grow

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

[2, 3].

The student–supervisor allocation problem is widely recognized as a constrained matching problem that must balance several competing criteria simultaneously: student research interests, supervisor expertise, ranked preferences, supervision capacity, fairness, and departmental policy [2, 4]. Consequently, a substantial body of research has explored algorithmic approaches to the problem, including integer linear programming [3], stable matching [6], weighted-product and simple additive weighting methods [7, 8], genetic algorithms [9], and particle swarm optimization [10]. These studies demonstrate that computational methods can represent allocation requirements more systematically than informal judgment.

A practical institutional system, however, requires more than an allocation algorithm. It must also provide the interfaces through which supervisors declare expertise and capacity, students submit preferences, administrators review and confirm outcomes, and all stakeholders receive timely notification. The present study addresses this combined software-engineering problem by designing and implementing a web-based Smart Project Balloting System (SPBS) that integrates weighted expertise matching and capacity-constrained workload management within an operational, role-based platform. In this paper, "workload management" denotes the enforcement of declared supervision capacities and the monitoring of supervisor utilization; it does not denote the minimization of an explicit workload-imbalance objective (Section 3.5).

1.2 Problem Statement

The core problem addressed in this study is the difficulty of producing fair and suitable student–supervisor assignments when several competing criteria must be considered at once. Manual allocation commonly leads to: (i) inconsistent, subjective decisions that are difficult to justify or audit; (ii) weak alignment between a student's project interests and the supervisor's expertise; (iii) uneven distribution of students, with some supervisors overloaded and others underutilized; and (iv) fragmented records spread across paper files, spreadsheets, and messaging platforms, which complicates updates, appeals, and longitudinal analysis. Although previous research provides mathematical and algorithmic solutions to allocation problems [2, 4, 5], there remains a need for a deployable platform that combines matching logic with data collection, persistent storage, administrative oversight, and communication in a single workflow.

1.3 Aim and Objectives

The aim of this study is to design and implement a web-based Smart Project Balloting System that integrates weighted expertise matching with capacity-constrained workload management to support more transparent, consistent, and efficient project allocation. The specific objectives are to:

- design a role-based web platform serving administrators, supervisors, and students;
- capture supervisor expertise and capacity declarations, project-topic proposals, student research interests, and ranked preferences;
- implement a weighted matching process that combines expertise compatibility with preference information;
- enforce declared supervision-capacity limits during allocation and report workload distribution through explicit utilization metrics;
- provide administrative review, result publication, automated notification, and reporting functions; and
- evaluate the implemented prototype through structured functional testing and a simulation-based assessment of allocation quality against random and first-come balloting baselines.

1.4 Scope of the Study

The system covers registration and authentication, supervisor expertise and capacity management, project-topic submission and approval, student research-interest and preference submission, automated matching under capacity constraints, administrative review and publication of assignments, notification, and reporting. Human factors such as personality and communication compatibility are outside the scope of the matching model, and the evaluation is limited to functional testing of the prototype rather than long-term measurement of

supervision outcomes. The allocation-quality evaluation uses synthetic cohorts generated for this purpose, not real institutional records.

RELATED WORK

2.1 Algorithmic Approaches to Supervisor Allocation

A long tradition of research treats student–project and student–supervisor allocation as a formal combinatorial optimization problem. Early work formulated the assignment as integer programming models and explored stable-matching adaptations of the classical Gale–Shapley framework, establishing the theoretical foundation for automated allocation [2, 6]. Ramotsisi et al. [3] combined stakeholder surveys with an integer linear programming model implemented in OpenSolver, demonstrating measurable improvements in preference matching and workload balance in an engineering department while reducing the time burden on coordinators. Şimşek [5] proposed a multi-objective binary programming model for stable supervisor–student allocation in postgraduate programmes, implemented as a decision-support tool that lets administrators trade off student-oriented, supervisor-oriented, and jointly optimized allocation strategies. Yong and Salleh [13] examined final-year project allocation at Universiti Brunei Darussalam and proposed a linear programming approach aimed at an optimal and fair allocation of students to projects.

2.2 Decision-Support and Weighted-Criteria Methods

A complementary line of work applies multi-criteria decision-making methods to compute supervisor suitability indices. Arifin and Mintamanis [7] developed a desktop decision-support system based on the Weighted Product method, combining ten supervisor-selection criteria into a composite recommendation at Tadulako University, Indonesia. Asfani et al. [8] implemented a final-project information system using the Simple Additive Weighting algorithm, weighting factors such as academic rank, topic–expertise alignment, and current supervision load, and reported high validity across all user groups. More recent studies have introduced metaheuristic techniques: Serek and Zhaparov [9] applied a genetic algorithm incorporating preference satisfaction and psychological compatibility, while Rianat [10] combined support-vector classification, TF-IDF feature extraction, and particle swarm optimization for researcher–mentorship matching. Evidence from the education literature further confirms that stronger topic–expertise alignment is associated with higher-quality supervision experiences and more relevant feedback [1, 11], and that workload imbalance degrades the timeliness and quality of supervision [3]. A systematic review by Hussain et al. [12] shows that the practical effectiveness of any allocation method depends heavily on how well it is integrated into the administrative workflow of the institution.

2.3 Research Gap

The reviewed studies establish the computational nature of the allocation problem and demonstrate a range of viable optimization techniques. The gap addressed by this study is the practical integration of these considerations into a single, deployable, user-facing platform. The proposed system connects information collection (expertise, capacity, topics, preferences), weighted matching, capacity enforcement, administrative review and override, publication, notification, and audit-oriented reporting in one web application, positioning the allocation algorithm within the operational workflow of a university department rather than as a standalone offline model. Unlike the optimization-based formulations in [3, 5, 13], the present system does not claim to solve the assignment problem to optimality: it uses a lightweight, transparent heuristic and contributes an explicit specification of the scoring rule, an evaluation of that heuristic against baselines and an exact reference solution, and an analysis of its fairness and workload implications.

MATERIALS AND METHODS

3.1 Development Methodology

The system was developed using the Prototype Development Methodology, selected because the platform serves multiple stakeholder roles whose requirements are best elicited through early interaction with a working

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

system. The stages applied were: (1) requirements gathering and analysis, through review of the existing manual process; (2) quick design of the architecture, database schema, interface mockups, and matching framework; (3) construction of a working prototype of the core modules; (4) user evaluation and feedback collection; (5) iterative refinement of the interface and matching parameters; (6) development of the final integrated system; and (7) implementation and systematic testing.

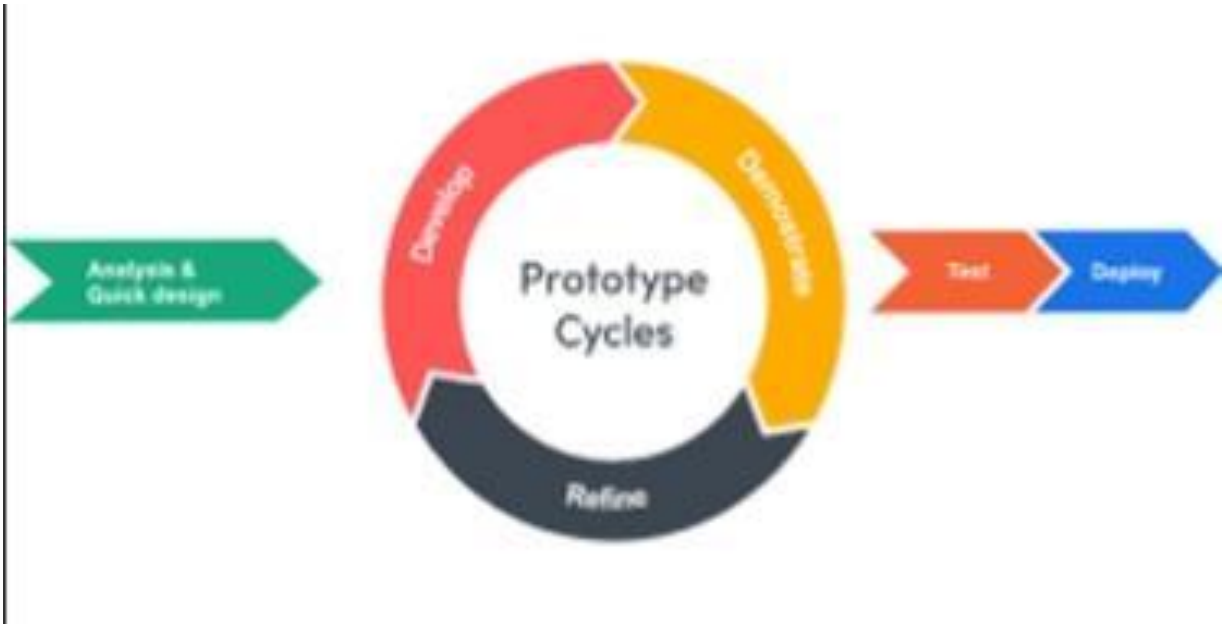


Figure 1: Prototype Development Methodology

3.2 Analysis of the Existing System

Under the existing manual process, students participate in a physical balloting exercise and subsequently select from available supervisors; the coordinator records assignments by hand, students submit project topics on paper or through informal channels, and results are communicated via notice boards, email, or messaging platforms. This approach exhibits several documented weaknesses: first-come-first-served bias in physical balloting; workload imbalance caused by the absence of systematic load tracking; poor topic–expertise alignment arising from informal assignment; risks of human error, duplicate entries, and data loss; high administrative effort; poor retrieval of historical records; and the absence of real-time status information for students and supervisors.

3.3 System Requirements

The application supports three roles. Administrators manage accounts, review and approve project topics, control submission windows, configure matching parameters, initiate the balloting process, review and adjust generated assignments, publish results, and generate reports. Supervisors declare expertise areas and maximum supervision capacity, propose and manage project topics, and view their assigned students. Students maintain profiles, browse approved supervisors and topics, submit ranked preferences, and view confirmed assignments. Principal non-functional requirements include password-based authentication with role-based access control, encrypted password storage, audit logging of allocation decisions and overrides, acceptable response times for the matching process, and interfaces usable on standard desktop and mobile browsers.

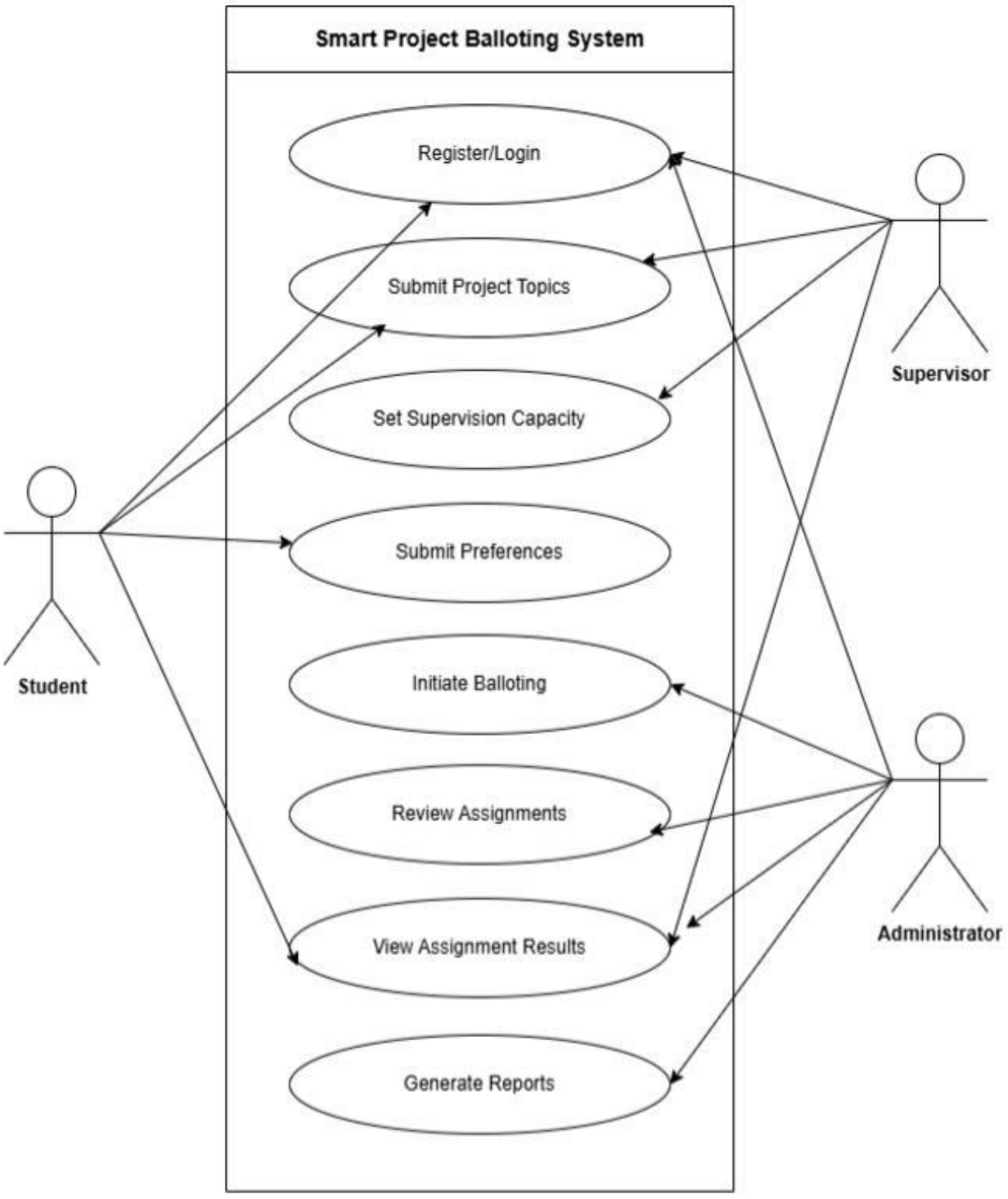
3.4 System Architecture and Workflow

The system follows a layered, object-oriented architecture comprising a presentation layer, an application layer, and a data layer. The presentation layer provides role-specific browser interfaces; the application layer implements authentication and authorization, topic and preference management, the weighted matching and capacity-checking logic, notification, and reporting; and the data layer persists users, supervisors, students, topics, preferences, and assignments in a relational MySQL database.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

The operational workflow proceeds as follows: the administrator opens the topic-submission window; supervisors declare expertise keywords, set capacity, and submit topics; the administrator reviews and approves topics; the preference window opens and students submit ranked choices; the administrator initiates balloting; the matching engine computes composite scores and generates assignments subject to capacity constraints; the administrator reviews the outcome, applies permitted manual adjustments, and confirms; the system then publishes the results and dispatches automated notifications to students and supervisors.

Figure 2: Use Case Diagram of the Smart Project Balloting System



ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

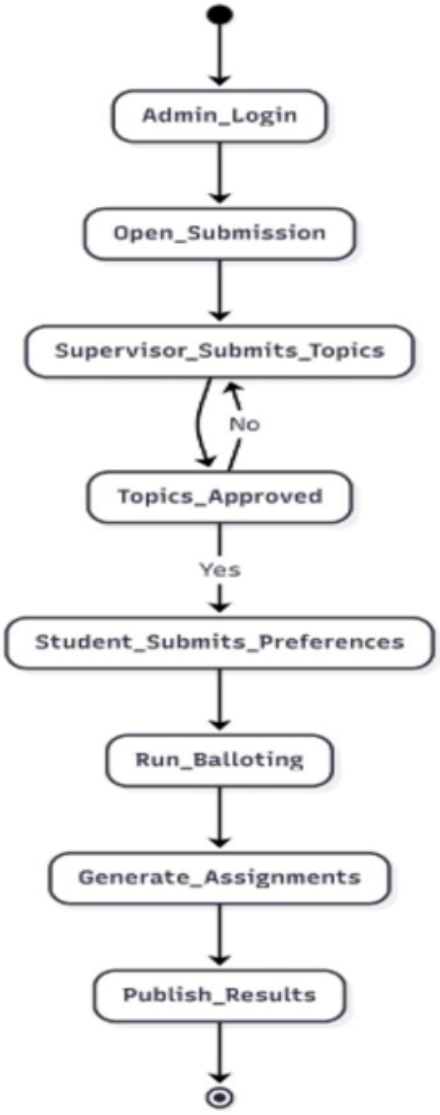


Figure 3: System Workflow/Activity Diagram

3.5 Weighted Matching and Capacity-Constrained Workload Management

3.5.1 Notation and Inputs

Let S = {s_1, ..., s_n} be the set of students and V = {v_1, ..., v_m} the set of supervisors. Each student s_i has a research-interest keyword set K_i; each supervisor v_j has a declared expertise keyword set X_j and a declared maximum capacity C_j. Each student submits a ranked list of at most R supervisors, and r(i, j) ∈ {1, ..., R} denotes the rank given by s_i to v_j (R = 5 in the evaluation reported in Section 4.4; R is configurable). The number of students assigned to v_j is denoted n_j.

3.5.2 Keyword Preprocessing and Expertise Score

Keywords are entered as free text separated by commas or semicolons and are preprocessed as follows: (i) conversion to lower case and removal of leading and trailing whitespace; (ii) removal of punctuation and of duplicate entries within a set; and (iii) exact matching of the resulting phrases. No stemming, synonym expansion, or fuzzy matching is applied, so "Machine Learning" and "machine learning" match, whereas "ML" and "machine learning" do not (a limitation discussed in Section 4.5). The expertise score is the fraction of the student's keywords covered by the supervisor's declared expertise:

$E(i, j) = |K_i \cap X_j| / |K_i|, \quad E(i, j) = 0 \text{ if } K_i \text{ is empty} \qquad (1)$

Normalizing by |K_i| rather than by |K_i ∪ X_j| (the Jaccard index) prevents supervisors from being penalized for declaring a broad expertise profile, and guarantees E(i, j) ∈ [0, 1], with E(i, j) = 1 when every student keyword is covered.

3.5.3 Preference Score

The preference score decays linearly with the rank assigned by the student, and is zero for supervisors the student did not rank:

$P(i, j) = (R - r(i, j) + 1) / R \text{ if } v_j \text{ is ranked by } s_i; \quad P(i, j) = 0 \text{ otherwise} \qquad (2)$

With R = 5, ranks 1 to 5 receive scores of 1.0, 0.8, 0.6, 0.4, and 0.2 respectively, so P(i, j) ∈ [0, 1] and the rank-1 supervisor always attains the maximum preference score.

3.5.4 Composite Score and Choice of Weights

The composite suitability score is:

$W(i, j) = \alpha \cdot E(i, j) + \beta \cdot P(i, j), \quad \text{where } \alpha + \beta = 1, \ \alpha, \beta \geq 0 \qquad (3)$

The weights α and β are policy parameters that determine how strongly the allocation favors topical fit over student choice; they are set by the administrator. The neutral default α = β = 0.5 is used in the main experiments because there is no a priori reason to privilege one criterion. The sensitivity analysis in Section 4.4 shows that the choice matters: moving α from 0 to 1 raises mean expertise compatibility from 0.36 to 0.69 but lowers first-choice satisfaction from 77.5% to 29.1%. Within 0.4 ≤ α ≤ 0.6, top-3 satisfaction remained at or above 84% while mean expertise compatibility remained at or above 0.56, which we therefore regard as a reasonable starting range; the final value should be validated with departmental stakeholders (Section 5).

3.5.5 Allocation Procedure, Tie-Breaking, and Fallback

Allocation uses a greedy ranking procedure, summarized in Algorithm 1. Candidate pairs with W(i, j) > 0 are ranked globally in descending order of score and assigned in that order whenever the student is still unassigned and the supervisor still has capacity.

Algorithm 1: Capacity-constrained greedy allocation with fallback

Input: E(i, j), P(i, j), capacities C_j, weight α (β = 1 − α), random seed σ

Output: assignment a(i) for each student, with score and method (primary or fallback)

1. Compute W(i, j) = α·E(i, j) + β·P(i, j) for all pairs; let Γ = {(i, j) : W(i, j) > 0}.
2. Draw a pseudo-random order π of the students using seed σ (stored in the audit log).
3. Sort Γ by: (a) W descending; (b) P descending; (c) position of the student in π; (d) supervisor ID.
4. Set n_j = 0 for all j and a(i) = none for all i.
5. For each (i, j) in sorted Γ: if a(i) = none and n_j < C_j, then a(i) = j and n_j = n_j + 1.
6. Fallback: for each student i with a(i) = none, taken in the order π:

among supervisors with n_j < C_j, select the one with the largest remaining capacity C_j − n_j

(ties: higher E(i, j), then lower supervisor ID); set a(i) = j, n_j = n_j + 1, and flag i as fallback.

7. If any student remains unassigned (possible only when Σ C_j < n), report these students to the administrator.
8. Return a, with W, E, P, method, and σ recorded for every assignment.

Ties. Ties in W(i, j) are common because E(i, j) takes only a few distinct values (multiples of 1/|K_i|) and P(i, j) takes R + 1 values. Ties are resolved first in favor of the pair with the higher preference score, so that student choice is respected when expertise is equal; then by the student's position in a seeded pseudo-random order, which avoids the first-come-first-served bias of physical balloting while keeping the result reproducible because the seed is stored in the audit log; and finally by supervisor ID so that the outcome is fully deterministic.

Fallback. A student is left unassigned by the primary pass only when every supervisor with $W(i, j) > 0$ for that student has reached capacity. The fallback routine then places the student with the supervisor having the most remaining capacity. Such students are, by construction, placed without regard to expertise or ranking, so the system flags them for administrative review during the override stage. If total declared capacity is at least n, every student is assigned and no capacity is exceeded; otherwise the unassigned students are reported (step 7).

Complexity and optimality. Sorting dominates the cost, giving $O(nm \log(nm))$ time for n students and m supervisors, which is negligible at departmental scale (Section 4.4 reports measured run times). The procedure is a heuristic: it guarantees a feasible, capacity-respecting assignment but does not guarantee the assignment of maximum total score. The size of this gap is quantified against an exact solution in Section 4.4.

3.5.6 Workload Management and Metrics

Supervision capacity is a hard constraint: for every supervisor v_j, $n_j \leq C_j$ at every step, so no published assignment can exceed a declared limit. The procedure contains no objective term that minimizes the imbalance of load across supervisors. The fallback favors supervisors with the most remaining capacity, which tends to even out the load of the students it places, but this is not an optimization of the overall distribution. For this reason the paper uses the term capacity-constrained workload management, meaning capacity enforcement together with monitoring, and does not claim workload optimization. Workload distribution is measured with the following metrics, computed over all supervisors: the utilization $u_j = n_j / C_j$; the standard deviation σ_u of u_j (lower values indicate more even utilization); Jain's fairness index $J = (\Sigma u_j)^2 / (m \cdot \Sigma u_j^2)$ [14], where $J = 1$ indicates identical utilization; the minimum and maximum utilization; and the number of capacity violations (which must be zero).

Fairness considerations. Because pairs are processed in global score order, students whose best pairs have high scores are served first, while students with low-scoring or conflicting pairs are served later from a reduced set of options. The procedure offers no guarantee for the worst-off student, does not produce a stable matching in the sense of Gale and Shapley (supervisors express no preferences over students), and is not strategy-proof. These properties are examined empirically in Section 4.4 and discussed critically in Section 4.5.

The principal system inputs are supervisor expertise keywords, declared capacities, approved project topics, student research interests, ranked preferences, and the weighting parameters α and β. The principal outputs are the assignment set with per-assignment match scores, workload distribution information, preference-satisfaction statistics, automated notifications, and audit records.

3.6 Database Design

The database centralizes all information required by the balloting process and is organized around six entities (Table 1): Users, Students, Supervisors, Project Topics, Preferences, and Assignments. The Users entity supports authentication and role management; Supervisors stores department, expertise keywords, declared capacity, and current load; Students stores the profile and research-interest keywords used as matching input; Project Topics records title, description, keywords, owning supervisor, and approval status; Preferences records each student's ranked choices; and Assignments records the final outcome, linking a student, supervisor, and topic together with the computed match score, the assignment method (automatic or manual), and the confirmation timestamp. The assignment entity thereby preserves a complete, auditable record of how each allocation decision was made.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

Table 1: Database entities and their representative fields.

Entity	Representative fields	Purpose
Users	user_id, name, email, password hash, role	Authentication and access control
Students	student_id, matric_no, department, research interests	Matching input
Supervisors	supervisor_id, department, expertise keywords, max_students, current_load	Capability and workload
Project Topics	topic_id, title, description, keywords, supervisor, status	Approved topic records
Preferences	preference_id, student, supervisor/topic, rank	Student preference data
Assignments	assignment_id, student, supervisor, topic, match_score, method	Final allocation outcome

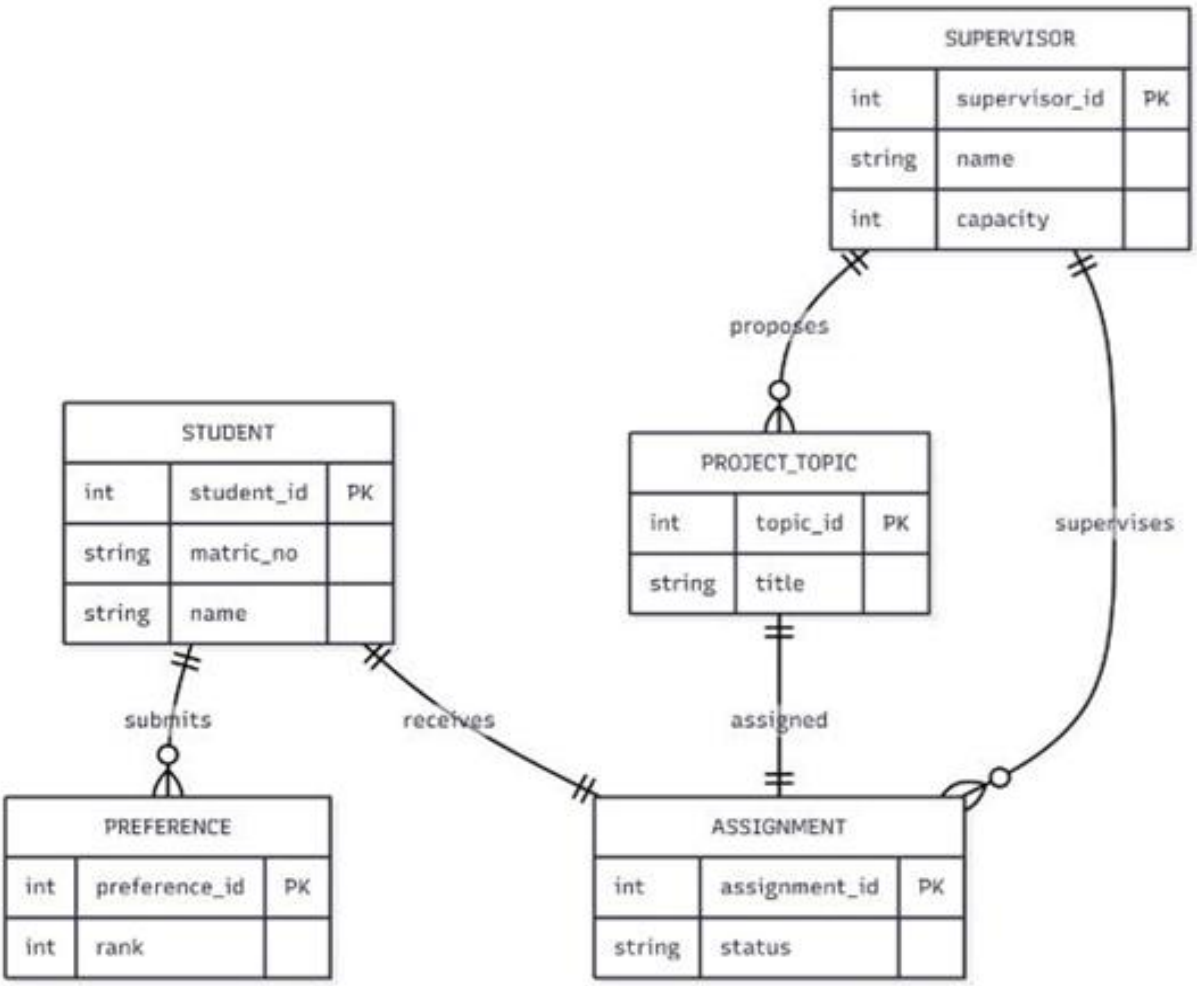


Figure 4: Entity Relationship Diagram

3.7 Evaluation Design

The seven functional tests reported in Section 4.3 verify that the software modules operate correctly but say little about the quality of the allocations produced. To assess the allocation strategy itself, a simulation study was conducted on synthetic cohorts, following the recommendation to report preference satisfaction, workload distribution, average matching score, unmatched students, execution time, and a comparison with simple baselines.

Synthetic cohorts. Each cohort contains n = 124 students and m = 18 supervisors, a scale typical of a single departmental cohort. A vocabulary of 42 keywords is divided into six research domains of seven keywords

each. Each supervisor declares five keywords from a primary domain and two from a randomly drawn secondary domain. Each student declares three to five keywords, each drawn from the student's primary domain with probability 0.8 and otherwise from the whole vocabulary. Each student ranks R = 5 supervisors, sampled without replacement with probability proportional to exp(2.5·E(i, j) + 0.8·ln(popularity_j) + noise), where popularity_j follows a skewed (Zipf-like) distribution; this represents students who favor topically fitting and popular supervisors and creates contention for places. Two capacity scenarios were generated: a tight scenario in which total declared capacity is about 1.05n, and a relaxed scenario in which it is about 1.30n. For each scenario 100 independent cohorts were generated using fixed seeds (1000 to 1099).

Methods compared. (1) *Random*: students, taken in random order, are placed with a randomly chosen supervisor that still has capacity. (2) *First-come balloting*, a model of manual physical balloting: students are taken in random draw order and each receives the highest-ranked supervisor that still has capacity, or a random free supervisor if none of the ranked ones remains; this baseline is driven by preference alone and ignores expertise. (3) *Proposed*: Algorithm 1 with $\alpha = \beta = 0.5$. (4) *Optimal reference*: the assignment maximizing the total score Σ W(i, j) subject to the capacity constraints, solved exactly as a linear assignment problem on capacity-replicated supervisor slots using the Hungarian-type algorithm [15] as implemented in SciPy [16]. The optimal reference is not a component of the system; it provides an upper bound on total score against which the greedy heuristic is judged.

Metrics. Preference satisfaction: the percentage of students receiving their first choice, one of their top three choices, and any ranked supervisor. Matching quality: mean expertise score E, mean composite score W, the percentage of students assigned a supervisor with no keyword overlap (E = 0), and total score as a percentage of the optimal reference. Completeness and feasibility: the number of unmatched students (after the primary pass and after the fallback) and the number of capacity violations. Workload distribution: σ_u, Jain's index J, and minimum and maximum utilization (Section 3.5.6). Efficiency: execution time of the allocation step. Results are reported as the mean ± standard deviation over the 100 cohorts. Because W is the quantity the proposed procedure ranks by, it favors that procedure by construction; E and the preference-satisfaction measures are therefore also reported as criteria that are not the procedure's own objective.

Scope. The simulation uses synthetic data, so absolute values depend on the generation parameters above and must not be read as predictions for a real department. The first-come baseline is a model of manual balloting, not observed practice.

IMPLEMENTATION AND RESULTS

4.1 Implementation

The system was implemented as a browser-based single-page application. The client side was built with React and TypeScript, with page routing handled by TanStack Router and the interface delivered as HTML5 and CSS3 rendered through React components; a server-side application layer implements the business logic and the matching engine; and a MySQL relational database provides persistent storage. The implemented modules comprise user registration, authentication, and role-based access control; supervisor topic, expertise, and capacity management; student research-interest and ranked-preference submission; the weighted matching engine with capacity enforcement; assignment generation; administrative review and permitted overrides; automated notification; and workload and preference reporting. Browser-based delivery allows all stakeholders to use the platform without installing specialized software, while the relational database provides structured, auditable storage of all allocation records.

4.2 User Interfaces

The implemented platform provides a separate interface for each stakeholder group. The landing page (Figure 5) presents the system purpose and entry points for login and registration. The student dashboard (Figure 6) displays the preference-submission window status, the student's submission status, and the balloting process timeline, and provides access to browse supervisors and approved topics, submit ranked preferences, and view the confirmed assignment. The supervisor dashboard (Figure 7) supports topic submission and editing,

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

expertise and capacity declaration, monitoring of topic approval status, and viewing of assigned students with current load relative to capacity. The administrator dashboard (Figure 8) provides user management, topic approval, control of submission windows, matching-parameter configuration, initiation of balloting, review of generated assignments with match scores, manual adjustment where institutional policy permits, result publication, and workload reporting. The results and reports page (Figure 9) presents the assignment outcome with summary match and preference statistics and a per-supervisor workload report, supporting transparent administrative review before publication. The screenshots were captured from the prototype populated with illustrative demonstration data; the figures displayed in them (for example cohort sizes and summary percentages) are not results of the evaluation reported in Section 4.4.

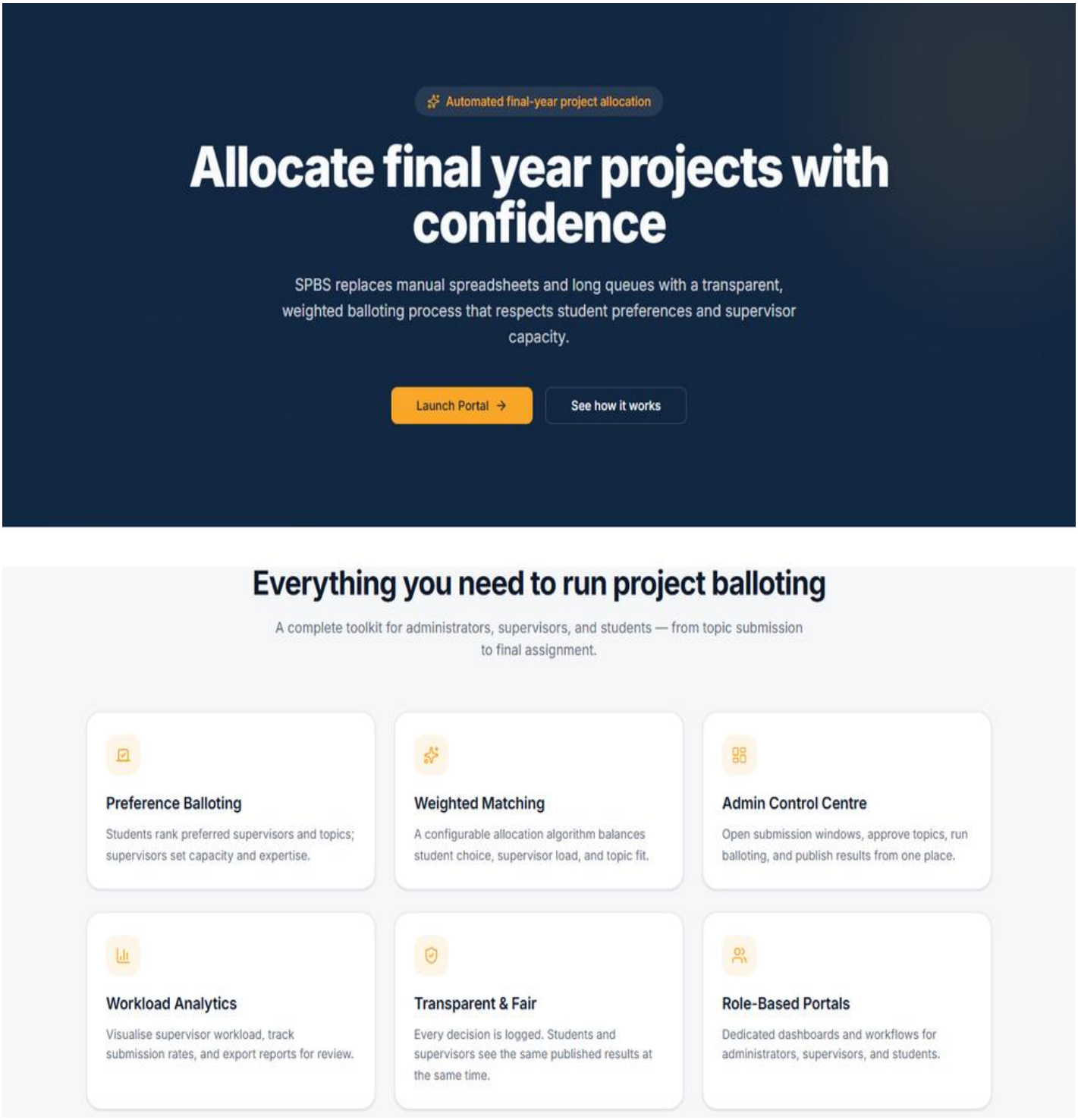


Figure 5: Landing page of the Smart Project Balloting System (hero and feature sections)

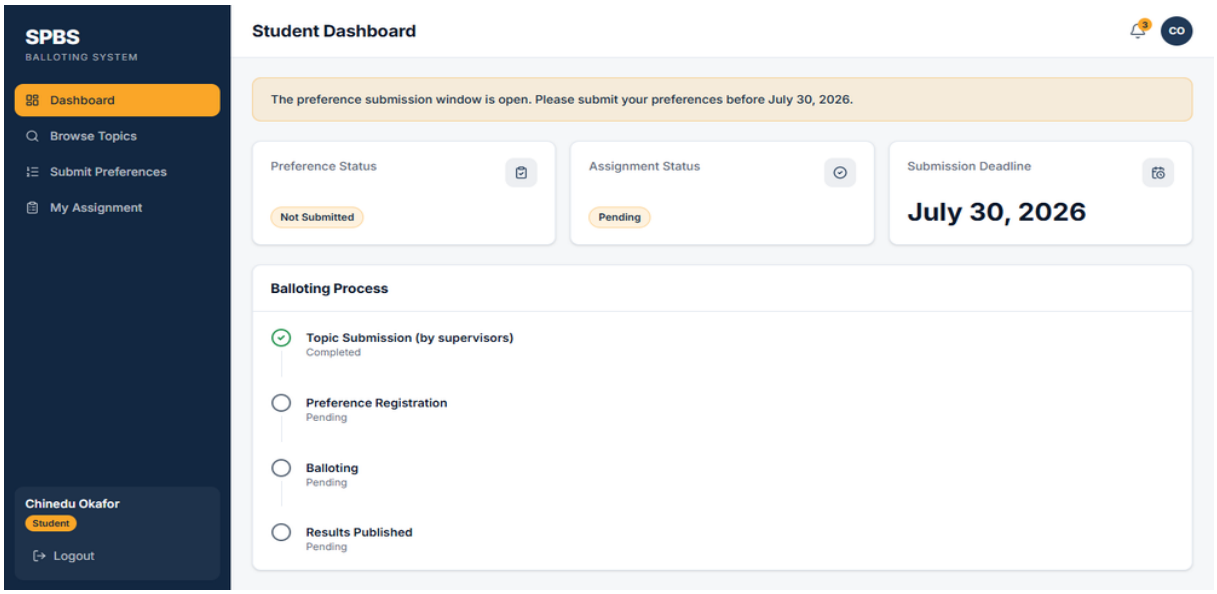


Figure 6: Student dashboard showing submission status and the balloting timeline

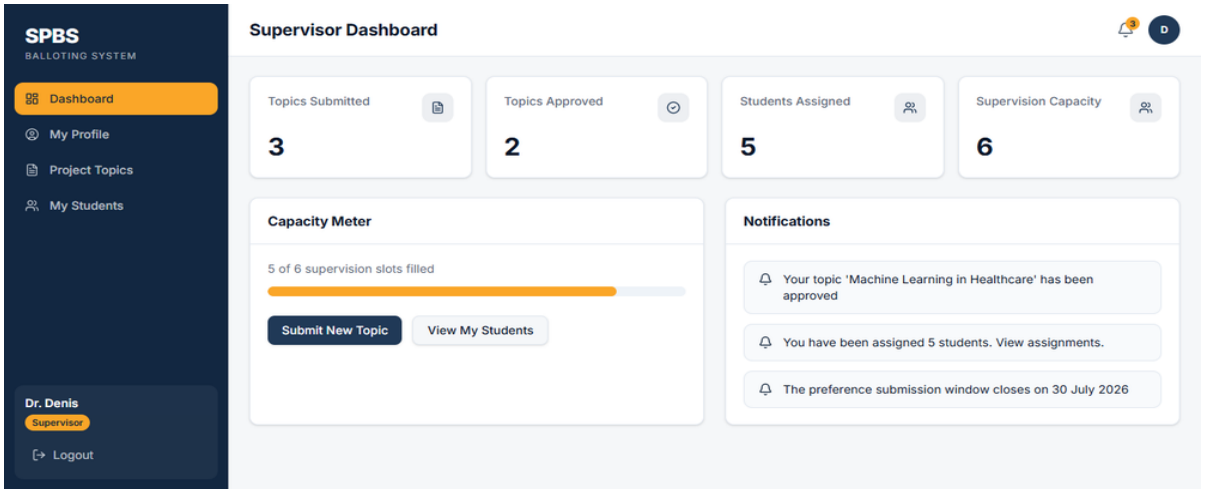


Figure 7: Supervisor dashboard showing topic status, assigned students, and the capacity meter

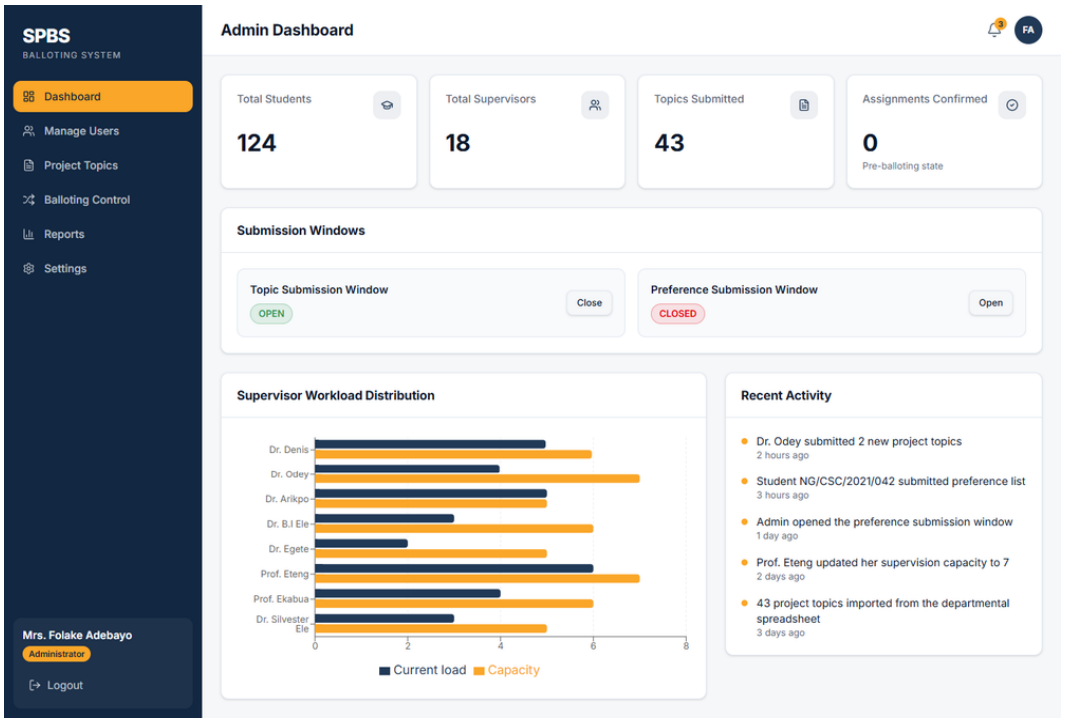


Figure 8: Administrator dashboard showing submission-window controls and supervisor workload against capacity (illustrative demonstration data)

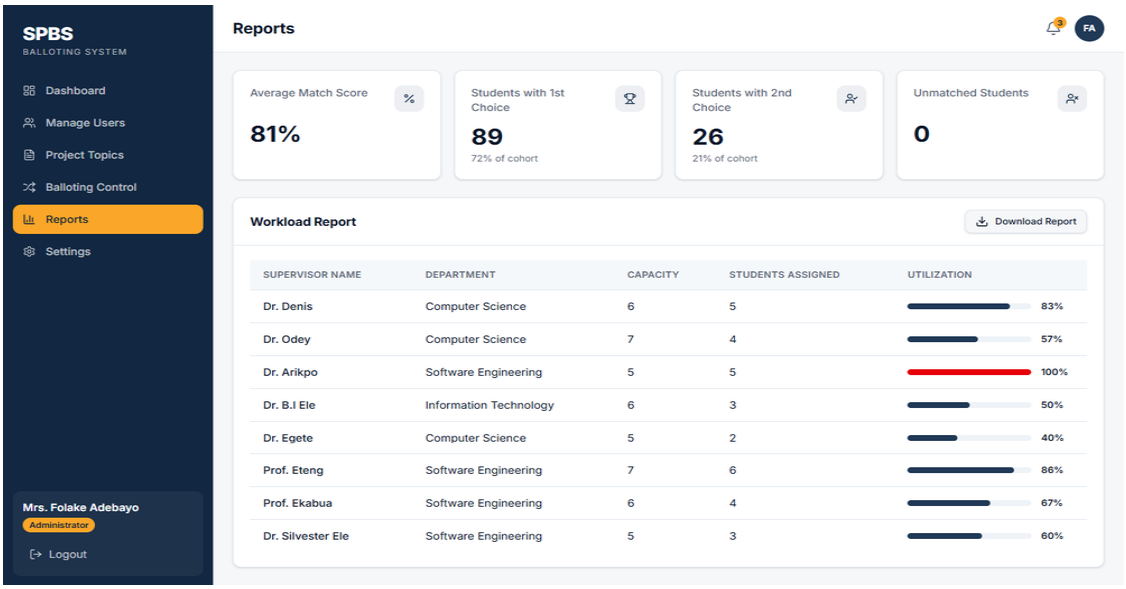


Figure 9: Balloting results and reports page with workload report (illustrative demonstration data)

4.3 Testing

The prototype was evaluated through structured functional (unit-level) testing of the principal modules. Seven documented test cases were executed, covering the full workflow from initialization to notification (Table 2); all seven passed.

Table 2: Functional test cases and results.

Test case	Function tested	Result
SPBS001	System initialization	Pass
SPBS002	User registration	Pass
SPBS003	User authentication	Pass
SPBS004	Supervisor topic submission	Pass
SPBS005	Student preference submission	Pass
SPBS006	Balloting engine execution	Pass
SPBS007	Assignment notification	Pass

In the balloting-engine test, all students in the test scenario received an assignment, no supervisor exceeded a declared capacity, and the generated results were correctly displayed on the respective dashboards. The notification module delivered confirmed outcomes to students and supervisors after administrative confirmation. These tests verify functional correctness only; they do not measure the quality of the allocations, which is examined in Section 4.4.

4.4 Allocation-Quality Evaluation by Simulation

Table 3 and Figure 10(a) compare the four methods under the tight-capacity scenario, and Table 4 reports the relaxed scenario (design in Section 3.7). All results are for synthetic cohorts.

Table 3: Tight-capacity scenario (total capacity ≈ 1.05n; 124 students, 18 supervisors; mean ± SD over 100 cohorts).

Metric	**Random**	**First-come balloting**	**Proposed (α = 0.5)**	**Optimal reference**
First choice (%)	5.2 ± 1.8	71.4 ± 5.5	52.5 ± 3.8	54.9 ± 4.2

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

Top-3 choice (%)	16.1 ± 3.2	91.8 ± 3.3	88.5 ± 3.1	93.4 ± 2.3
Any ranked supervisor (%)	27.1 ± 3.9	96.7 ± 2.2	95.0 ± 2.2	98.9 ± 1.0
Mean expertise score E	0.16 ± 0.02	0.35 ± 0.03	0.59 ± 0.04	0.59 ± 0.04
Mean composite score W	0.16 ± 0.02	0.62 ± 0.02	0.70 ± 0.02	0.72 ± 0.02
No keyword overlap, E = 0 (%)	65.8 ± 4.0	38.9 ± 4.9	12.2 ± 4.6	11.3 ± 4.6
Total score (% of optimal)	22.4 ± 2.5	85.5 ± 2.5	97.2 ± 1.0	100.0 ± 0.0
Unmatched students (final)	0	0	0	0
Capacity violations	0	0	0	0
Utilization SD, σ_u	0.10 ± 0.05	0.10 ± 0.05	0.09 ± 0.05	0.11 ± 0.06
Jain index, J	0.987 ± 0.010	0.985 ± 0.013	0.987 ± 0.011	0.984 ± 0.013
Minimum utilization (%)	66 ± 18	62 ± 20	68 ± 18	62 ± 21
Maximum utilization (%)	100 ± 0	100 ± 0	100 ± 0	100 ± 0

Matching quality and preference satisfaction. The proposed procedure produced markedly better expertise alignment than first-come balloting: mean expertise score 0.59 versus 0.35, and 12.2% versus 38.9% of students placed with a supervisor sharing no keyword with them. The price is lower preference satisfaction: 52.5% of students received their first choice, compared with 71.4% under first-come balloting, and 88.5% received one of their top three choices, compared with 91.8%. The first-come baseline attains high preference satisfaction because it is driven by preference alone, whereas the proposed procedure deliberately trades part of it for expertise alignment. The mean composite score is also higher for the proposed procedure (0.70 versus 0.62), but, as noted in Section 3.7, this is the quantity the procedure ranks by. Both methods far outperformed random placement.

Completeness, feasibility, and efficiency. No student was left unmatched and no capacity limit was exceeded in any cohort. The primary pass left on average 1.97 students (SD 1.66; about 1.6% of the cohort) to be placed by the fallback in the tight scenario, and 0.06 in the relaxed scenario. Allocation took 11 ms on average (SD 3 ms) in the reference Python implementation for a cohort of this size; this measures the matching step only and not end-to-end coordinator effort.

Gap to the optimal solution. The greedy procedure reached 97.2% (SD 1.0; worst cohort 94.9%) of the optimal total score in the tight scenario and 99.0% (worst cohort 97.2%) in the relaxed scenario. The exact solution also gave slightly higher top-3 satisfaction (93.4% versus 88.5%) and a similar share of students with no keyword overlap (11.3% versus 12.2%).

Workload distribution. Utilization dispersion under the proposed procedure (σ_u = 0.09, J = 0.987) was essentially the same as under first-come balloting (σ_u = 0.10, J = 0.985) and random placement (σ_u = 0.10, J = 0.987). With ample spare capacity (Table 4) dispersion was larger for all methods, and the least-utilized supervisor was filled to only 27% on average while others reached 100%. The procedure therefore respects capacity but does not improve utilization balance over the baselines; workload distribution is governed mainly by the declared capacities and the pattern of student demand.

Table 4: Relaxed-capacity scenario (total capacity ≈ 1.30n; mean ± SD over 100 cohorts).

Metric	**Random**	**First-come balloting**	**Proposed (α = 0.5)**	**Optimal reference**
First choice (%)	5.4 ± 1.8	83.1 ± 4.6	56.0 ± 4.0	57.8 ± 4.6
Top-3 choice (%)	16.1 ± 3.0	98.6 ± 1.4	93.6 ± 2.1	95.1 ± 2.0
Mean expertise score E	0.16 ± 0.02	0.37 ± 0.03	0.60 ± 0.04	0.60 ± 0.04
No keyword overlap, E = 0 (%)	66.2 ± 4.2	37.4 ± 4.2	10.1 ± 3.8	9.4 ± 3.8
Total score (% of optimal)	21.7 ± 2.7	89.4 ± 1.7	99.0 ± 0.5	100.0 ± 0.0
Utilization SD, σ_u	0.21 ± 0.03	0.24 ± 0.02	0.24 ± 0.03	0.24 ± 0.03

Jain index, J	0.929 ± 0.016	0.912 ± 0.017	0.910 ± 0.020	0.910 ± 0.020
Minimum utilization (%)	34 ± 10	27 ± 10	27 ± 10	26 ± 11

Sensitivity to the weights. Table 5 and Figure 10(b) show the effect of α on the proposed procedure in the tight scenario (50 cohorts per setting). Increasing α raises mean expertise compatibility throughout and lowers first-choice satisfaction throughout; top-3 satisfaction is nearly flat up to α = 0.4 and falls steeply beyond α = 0.6. Utilization dispersion is almost unaffected by α. The fallback was needed for more students at the two extremes: at α = 1 (expertise only) because pairs with no keyword overlap are not candidates, and at α = 0 (preference only) because students whose ranked supervisors are all full have no other candidates.

Table 5: Effect of the expertise weight α on the proposed procedure, tight-capacity scenario (mean over 50 cohorts).

α	First choice (%)	Top-3 (%)	Mean E	No overlap (%)	Utilization SD σ_u	Fallback students
0	77.5	91.8	0.36	37.6	0.08	5
0.2	75.5	91.8	0.45	26.1	0.09	1.4
0.4	58.4	90.6	0.56	15.1	0.09	1.9
0.5	52.3	88.5	0.59	12.1	0.09	2
0.6	46.2	84.5	0.62	9.4	0.09	1.9
0.8	34.5	68.2	0.68	5.7	0.09	2.4
1	29.1	60.6	0.69	5.2	0.07	6.4

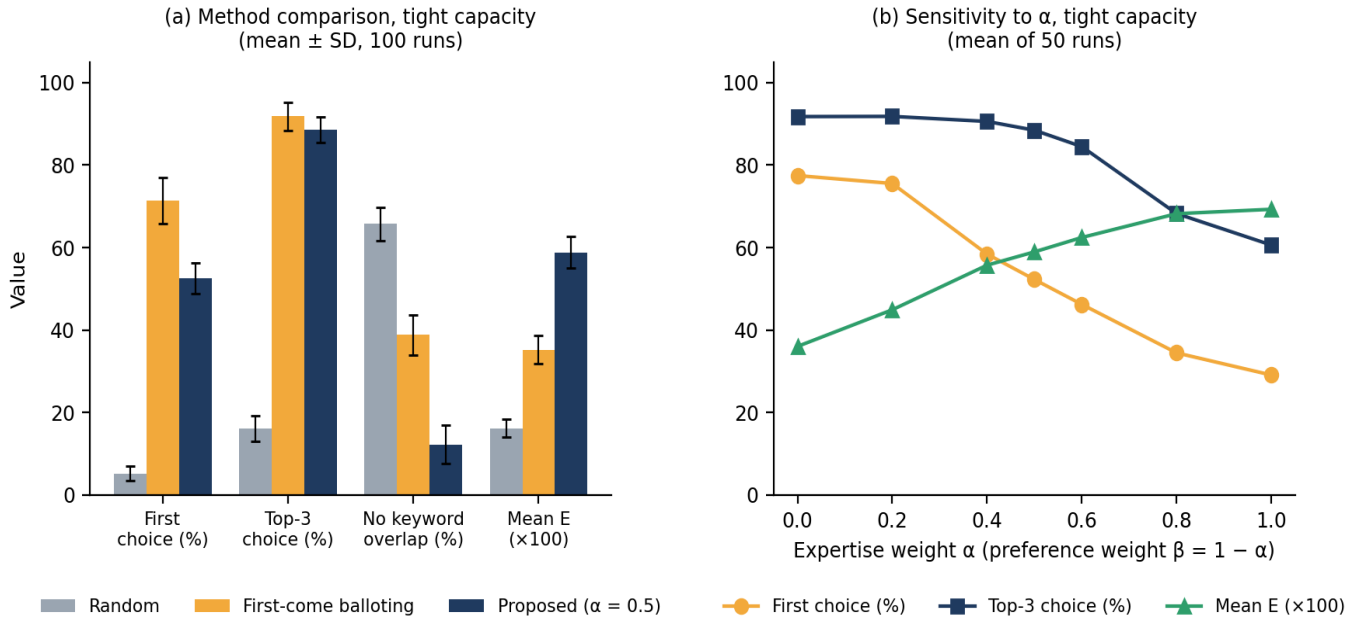


Figure 10: (a) Method comparison and (b) sensitivity to the expertise weight α in the tight-capacity scenario (synthetic data)

4.5 Discussion

The functional tests demonstrate the feasibility of the proposed system: the complete workflow, from supervisor topic submission through preference collection, weighted matching, capacity-constrained assignment, administrative review, and notification, executes successfully and produces auditable records. By replacing informal judgment with an explicit, configurable scoring rule (α and β), a seeded and logged tie-

breaking rule, and hard capacity limits, the system makes the allocation rule inspectable and repeatable, while preserving administrative authority through the review-and-override stage. This human-in-the-loop design is consistent with recommendations in the literature that allocation systems should keep decision-makers accountable and outcomes explainable [5, 12]. These properties of transparency and consistency follow from the design of the system; they were not measured empirically in this study.

Allocation quality and the expertise-preference trade-off. The simulation shows that the weighted score does what it is designed to do: it substantially increases topical alignment (mean E 0.35 to 0.59; students with no keyword overlap 38.9% to 12.2%) compared with preference-driven first-come balloting. It does not make allocations uniformly better: first-choice satisfaction fell by about 19 percentage points. Whether this exchange is desirable is a matter of departmental policy, which is why α is administrator-configurable, and the sensitivity analysis gives administrators an indication of the consequences of each setting. The comparison with a model of manual balloting on synthetic data should not be read as evidence that the system outperforms real coordinators.

Fairness and the greedy heuristic. Global greedy ranking is simple and explainable, and in the simulation it obtained 97.2% of the optimal total score under tight capacity, so little total score is lost relative to an exact solution. Total score, however, is an aggregate and says little about how outcomes are distributed across students. Roughly 12.2% of students were still placed with a supervisor sharing none of their keywords and about 1.6% needed the fallback, and the procedure offers no guarantee for the worst-off student: students whose best pairs score low are served late, from a reduced set of options. The procedure is also not a stable matching and not strategy-proof. In practice the administrative review stage is the main safeguard, since fallback assignments and low-score assignments are flagged for inspection. Egalitarian objectives (for example maximizing the minimum score), stable-matching formulations, and integer linear programming [3, 13] are natural alternatives that would address these concerns at greater computational and explanatory cost.

Workload. The simulation does not support a claim of workload optimization: utilization dispersion under the proposed procedure was comparable to that of random and first-come placement. The system enforces capacity and reports utilization, which is what the manuscript now claims. Achieving a more even distribution would require adding an explicit balancing term (for example penalizing the current utilization of a supervisor in the pair score) or a min-max load objective in an exact formulation, which is left to future work.

Limitations. First, the allocation-quality evidence comes from synthetic cohorts, so absolute figures depend on the generation assumptions in Section 3.7, and the first-come baseline is a model rather than observed manual practice; no controlled comparison with real allocations was available. Second, E is computed from exact keyword matches and therefore cannot recognize synonyms, abbreviations, paraphrases, or interdisciplinary relationships between topics; natural-language processing techniques would address this limitation. Third, the efficiency of the matching step was measured, but the end-to-end time saved by coordinators was not. Finally, long-term supervision outcomes after assignment are outside the scope of the present evaluation. Claims concerning transparency, consistency, and efficiency should accordingly be read as design properties and feasibility results rather than as experimentally demonstrated improvements.

CONCLUSION

This study presented the design and implementation of a Smart Project Balloting System that integrates weighted expertise matching, ranked student preferences, and supervision-capacity constraints within a single web-based institutional workflow. The system provides role-based access for administrators, supervisors, and students; captures the inputs required for multi-criteria allocation; enforces capacity limits during assignment; and supports administrative review, publication, automated notification, and reporting. Seven documented functional test cases covering the major modules all passed, demonstrating that the platform operates correctly end to end. A simulation of 100 synthetic cohorts showed that, relative to a model of first-come balloting, the weighted procedure substantially improved expertise alignment (mean expertise score 0.35 to 0.59) at the cost of lower first-choice satisfaction (71.4% to 52.5%), reached 97.2% of the optimal total score, and never exceeded a capacity limit; it did not, however, improve utilization balance over the baselines.

The study contributes an implementable decision-support framework that connects allocation criteria to the administrative processes required for project balloting, together with an explicit specification of the scoring rule and an evaluation of its trade-offs. It does so without overstating the evidence: functional feasibility and the behavior of the heuristic on synthetic data are established, while performance on real cohorts and against actual manual allocation remains to be demonstrated. Future work will evaluate the system on real, anonymized institutional datasets; quantify preference satisfaction and workload distribution against a manual baseline; validate the weighting parameters with stakeholders; add an explicit workload-balancing objective and fairness-oriented allocation criteria; and extend the matching model with semantic (natural-language processing) techniques for research-interest comparison, mobile access, and stronger security controls.

ETHICAL CONSIDERATIONS

This study concerns the design and functional testing of an academic software prototype. No human-subject experiment and no personally identifiable dataset are reported. If institutional student or staff data are used in future evaluations, appropriate institutional authorization, privacy safeguards, and any required ethical approvals should be obtained.

CONFLICT OF INTEREST

The authors declare no conflict of interest in relation to this study.

DATA AVAILABILITY

The system is implemented as a web application. The data underlying the reported functional tests are not released as a public dataset, and no private student or staff data accompanies this manuscript. The simulation reported in Section 4.4 uses synthetic data generated by a seeded script (seeds 1000 to 1099), which is available from the corresponding author on request.

REFERENCES

1. Santos, A. (2025). Improving student-supervisor matching: Educational gains from a data-driven approach. Adam Smith Business School, University of Glasgow.
2. Gani, M. A., & Hamid, R. A. (2021). Optimum allocation of graduation projects: Survey and proposed solution. Journal of Al-Qadisiyah for Computer Science and Mathematics, 13(1), 58–74.
3. Ramotsisi, J., Kgomotso, M., & Seboni, L. (2022). An optimization model for the student-to-project supervisor assignment problem: The case of an engineering department. Journal of Optimization, 2022, Article 9415210.
4. Sanchez-Anguix, V., Chalumuri, R., Aydoğan, R., & Julián, V. (2019). A near Pareto optimal approach to student–supervisor allocation with two-sided preferences and workload balance. Applied Soft Computing, 76, 1–15. https://doi.org/10.1016/j.asoc.2018.11.049
5. Şimşek, A. B. (2022). A decision support tool for the student–supervisor allocation problem of postgraduate programs. Expert Systems with Applications, 190, Article 116068.
6. Manlove, D. F., & O'Malley, G. (2008). Student-project allocation with preferences over projects. Journal of Discrete Algorithms, 6(4), 553–560.
7. Arifin, S. R., & Mintamanis, J. C. (2018). Decision support system for determining thesis supervisor using a weighted product (WP) method. Jurnal Online Informatika, 3(2), 80–85.
8. Asfani, K., Firdaus, A. F., & Basyirun, B. (2022). Simple additive weighting algorithm implementation for determining supervisors in student final project information system. Letters in Information Technology Education, 5(1), 25–28.
9. Serek, A., & Zhaparov, M. (2024). Optimizing preference satisfaction with genetic algorithm in matching students to supervisors. Applied Mathematics & Information Sciences, 18(1), 133–138.
10. Rianat, A. O. (2024). Optimizing researcher mentorship matching: A particle swarm optimization-based recommendation model. WORLD, 13(2), 698–707.

11. Hunter, A. B., Laursen, S. L., & Seymour, E. (2007). Becoming a scientist: The role of undergraduate research in students' cognitive, personal, and professional development. Science Education, 91(1), 36–74.
12. Hussain, S., Gamage, K. A. A., Sagor, M. H., Tariq, F., Ma, L., & Imran, M. A. (2019). A systematic review of project allocation methods in undergraduate transnational engineering education. Education Sciences, 9(4), 258. https://doi.org/10.3390/educsci9040258
13. Yong, S. Y., & Salleh, N. Q. H. M. (2023). Evaluating effectiveness of student final year project allocation: A case study at Universiti Brunei Darussalam. International Journal for Multidisciplinary Research, 5(3), 1–7.
14. Jain, R., Chiu, D., & Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems (Technical Report DEC-TR-301). Digital Equipment Corporation.
15. Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1–2), 83–97.
16. Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 17, 261–272.