SmartAllot: Minor and OE Allocation System

Sahil Patil[1], Abhishek Chaudhri[2], Prathmesh Deshmukh[3], Anjali Tak[4], Mamta Gaykwad[5]
[1,2,3,4,5] *Information Technology Dept, Mauli Group of Institution's College of Engineering & Technology, Shegaon*

***Abstract* -: In colleges and universities, is important for administrators to help students choose minor specializations and open elective courses. Traditional manual allocation methods are time consuming, often errors prone, and are not easy to understand. This paper presents SmartAllot, an automatic system that uses merit to allocate minor branches and open elective courses in a way that makes the distribution more efficient, fair, and open. The system uses a method that ranks students based on their preferences in a set order of merit, checks if they meet the necessary requirements, ensures there are enough spots available, and keeps thorough records for tracking and review. SmartAllot uses Django 5.0 and runs on a PostgreSQL database. It allows multiple tenants to use it, and each tenant can have different roles with specific access rights. The institution gave real student information from 68 students in various departments to check if the system works properly. All 68 students got placed in a branch that aligned with one of their preferred choices. This meant every student was assigned properly and all the rules were followed. Allocation was done in under five seconds, whereas coordinators used to spend several days doing it by hand.**

Keywords: merit-based allocation, minor specializations, open electives, automation, Django framework, deterministic algorithm, eligibility validation, educational administration, audit trail, and role-based access control.

I. INTRODUCTION

Every semester, colleges and universities figure out which students will enroll in which classes. Handling many complex rules is needed when choosing minor specializations and open elective courses. These rules mean giving priority to those who are most qualified; ensuring that preferences are considered but not more than the allowed number of seats; checking that only eligible candidates are selected based on their branch and academic standing; making sure all decisions are transparent and follow the rules; and ensuring that all processes can be reviewed and checked. Using old manual spreadsheets isn't efficient, reliable, or clear. They take a lot of time and sometimes break the rules. SmartAllot was created because of the challenges faced at the College of Engineering and Technology in many institutions. There, people use manual spreadsheets and make decisions on the spot to handle allocations for various minor branches and open electives each semester. The main contributions of this work are: (1) a rank-based allocation system that determines student rankings based on their merit, preference order, and when they submitted their application; (2) a complete framework to check if students are eligible based on their academic standing and any branch limitations; (3) a detailed logging system that keeps track of all actions for full transparency and to ensure compliance; and (4) a fully functional Django 5.0 application that uses PostgreSQL as its database and includes role-based access control for security.

II. LITERATURE SURVEY

1. Gale and Shapley (1962) developed the classical college admissions and stable matching algorithm, establishing the fundamental theory for preference-based allocation. The deferred acceptance algorithm shows that there are always stable matches in two-sided markets. This is the mathematical basis for modern course allocation systems. SmartAllot uses the main idea that processing in order of merit leads to stable, fair results. [1]

2. Irving, Manlove, and Scott (2003) adapted the classical model to the hospitals/residents problem, incorporating ties and robust stability. Their research yields efficient $O(a^2)$ algorithms for identifying strongly stable matchings in situations where participants exhibit indifference, particularly applicable to cases with multiple students possessing identical academic scores. The multi-criteria tie-breaking (marks, timestamp, roll number) used by SmartAllot is based on the stability definitions that were created in this

work. [2]

3. Abraham, Irving, and Manlove (2007) examined the student-project allocation problem, an extension of the hospitals/residents problem, characterized by capacity constraints for both projects and lecturers, as well as lecturers' preferences regarding students. Their two best linear-time algorithms for finding stable matchings in this model helped SmartAllot design its allocation engine, especially the student-optimal stable matching method. [3]

4. Che and Kojima (2010) established the asymptotic equivalence of probabilistic serial and random priority mechanisms in extensive markets, illustrating that the inefficiency of random priority mechanisms diminishes as market size increases. This finding offers theoretical validation for SmartAllot's deterministic merit-based (priority) system: in reality, for institutions with hundreds of students, the merit-priority mechanism approaches near-optimal efficiency. [4]

5. Manlove (2013), in his extensive work Algorithmics of Matching Under Preferences, examined algorithmic and complexity outcomes for various matching issues, such as student-to-course allocation, kidney exchange, and hospital-resident pairing. This work gave SmartAllot's eligibility constraint modelling a bigger algorithmic context and made sure that an institutional system would be formally correct. [5]

6. In their book Database System Concepts, Silberschatz, Korth, and Sudarshan (2020) give an authoritative guide to relational database design, indexing strategies, and transaction management. The principles laid out in this foundational text directly influenced SmartAllot's PostgreSQL schema design, strategic B-tree indexing on marks and allocation status fields, and ACID-compliant allocation transactions. [6]

III. PROBLEM STATEMENT

There are five major issues with how engineering schools currently handle open elective courses and minor specializations: (1) Using a spreadsheet to manage resources manually makes it easy to make errors and follow rules inconsistently. (2) The process lacks transparency—students don't know or can't question how slots are allocated, and there's no record to check later. (3) The system works well only for up to 200–300 students. (4) It's difficult to check and apply complex rules like branch restrictions, backlog status, and CGPA limits by hand, which causes many mistakes. (5) There's no automatic way to handle students who don't submit their preferences on time.

IV. EXISTING SYSTEM AND ITS LIMITATIONS

The current way of allocating seats has five steps that are all done manually: first, students send in their choices either by email or on paper. Then, someone creates merit lists by hand using Excel. Next, the data is processed manually with VLOOKUP formulas, and this step is often done again if there are problems. After that, the data is checked by hand, and any issues or disagreements are fixed. Finally, the results are shared by printing them out or sending them through email. The technology used has Microsoft Excel along with VBA macros and a spot to keep CSV files. Traditional systems struggle with scaling, which is a major challenge. Branch restrictions and backlog statuses aren't automatically enforced. There is no system in place for setting up different permission levels based on roles and keeping track of who is responsible for what. People usually rely on manual methods, which can easily lead to errors. Version control is nonexistent. Data silos happen because different databases aren't connected together. There is no audit trail, and you can't get real-time reports.

V. PROPOSED SYSTEM

SmartAllot is a system you use on the web to handle how things are allocated. It uses Django 5.0 and PostgreSQL for its database. There are four layers: the Presentation Layer, which uses HTML5, CSS3, Bootstrap 5, and Django Templates for three different portal types based on user roles; the Application Layer, which includes Django view controllers, middleware, and RESTful URL routing; the Business Logic Layer, which has components like Allocation Engine, Eligibility Validator, Preference Manager, ReportGenerator, and Audit Logger; and the Data Persistence Layer, which uses PostgreSQL 13 or newer through the Django ORM.

Some key features include: (1) A system that allocates based on merit in a predictable way, using multiple criteria where the main factor is marks, then the time of the submission, and finally the roll number; (2) A flexible set of rules that control who is eligible, including limits on the branch they're in, their academic standing, and their CGPA; (3) A detailed record of all decisions made during the allocation process, including when each

decision was made and the reason behind it; and (4) Different access areas for students, coordinators, and administrators based on their roles.

VI. ALLOCATION ALGORITHM

SmartAllot's allocation process happens in three simple steps. First, based on their grades, each student is placed in order from the top performer to the lowest performer. The student who picked their options first will get a better position if two students have the same grades. If there is still a tie, the student with the lowest roll number wins. Before the process starts, this step makes sure that each student's place in the line is available.

Second, this is the order in which the system works with each student. Each student's top preferred branch is examined. It checks if the student meets the requirements needed for that course. For example, it checks if their CGPA is high enough, if their academic standing is clear, and if their department isn't limited to that specific field. The student is placed in that branch, and the number of seats goes down by one if they are qualified and there are still seats left in that branch. If a student is eligible but the branch has no more spots, they are put on the waitlist. The system goes to the next option and checks again if it doesn't meet the requirements. A student is placed in the first branch that has a seat available.

Third, the students who didn't submit their preferences are placed in an absconding phase after all the students who gave preferences have been assigned. They are put right into the first branch that they are eligible for, as long as there are still spots available there.

This algorithm takes O(N log N) time, with N being the number of students. The most time-consuming part is organizing the data. Even if more students join, the process stays very fast because checking each student's choices and qualification requirements always takes the same amount of time.

ALGORITHM PSEUDOCODE:

Output:

Last list of allocations A

List W of people waiting

1. Put all the students in order from highest to lowest marks.

If two students have the same grades, give priority to:

a) a shorter time to submit

b) a lower roll number

2. For each student:

Go through the student's preferences one by one.

If the student is qualified and there are seats available, put them in that branch and lower the number of seats.

3. If there aren't any open seats in a student's preferred branch, put them on the waiting list for that branch.

4. Students who didn't say what they wanted are put in the first branch with open seats and are eligible.

VII. TECHNOLOGIES USED

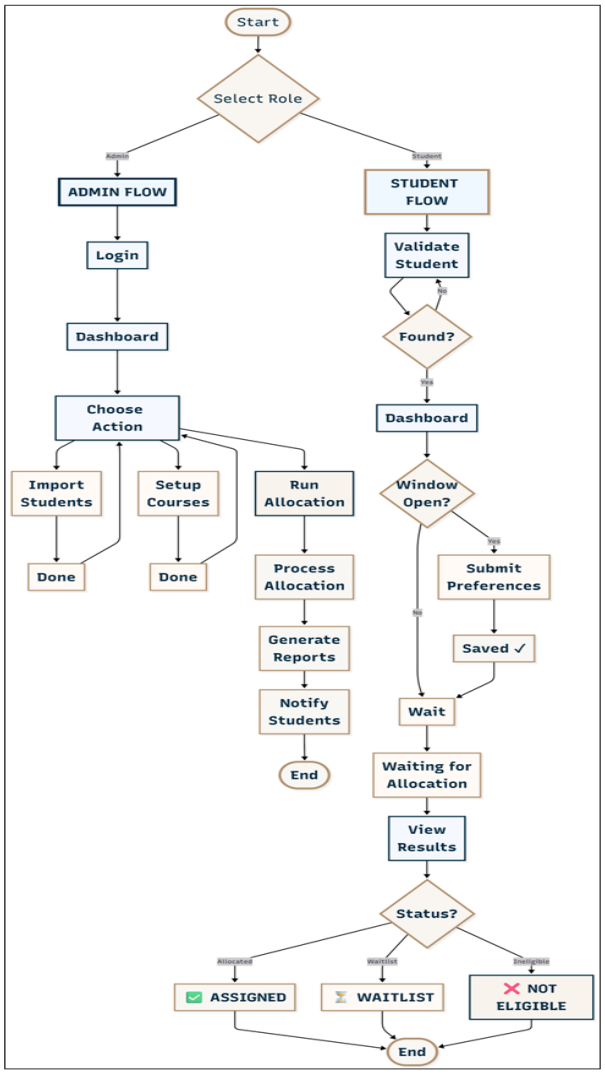
Technology Stack:

Frontend: HTML, CSS, JavaScript

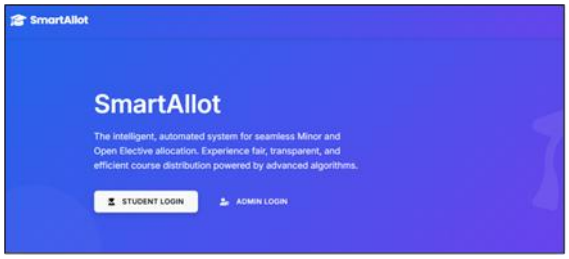
Backend: Python 3.10+, Django 5.0, Django REST Framework 3.15.1.

Database: PostgreSQL 13+ with Django ORM.

VIII. IMPLEMENTATION

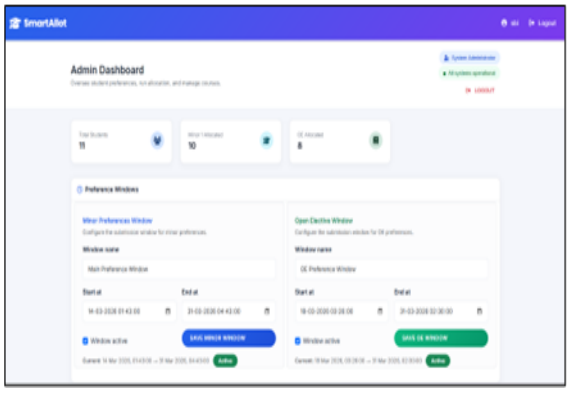


Landing Page:
The first page that staff and students see when they get there. It has information about the system, how different types of users (students, coordinators, and admins) can log in, and links to important features like the status of preference submissions and allocation results.

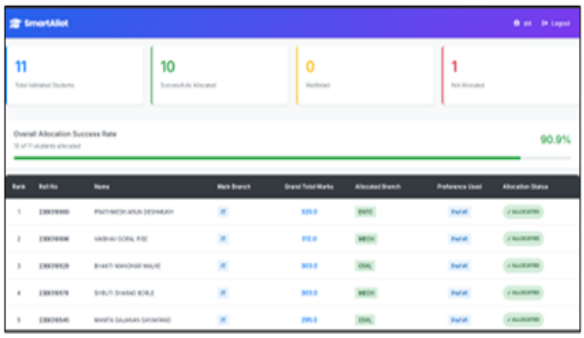


Admin Login Page:
On a secure login screen, users type in their username and password. The system checks your identity and sends you to the right dashboard for your job. Students go to their preference portal, while administrators go to the control panel.

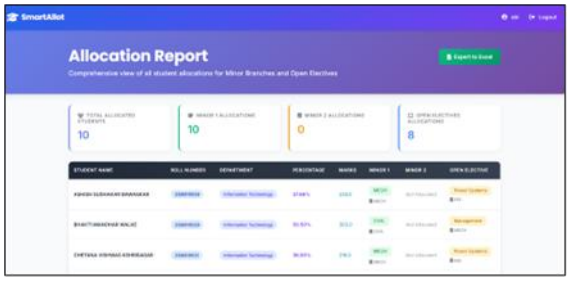
Admin Dashboard page:
This is where administrators do most of their work and manage the whole allocation process. It shows important information like the total number of students processed, the current status of allocations, open tasks, and buttons that let you quickly start allocations, set rules, and make reports.



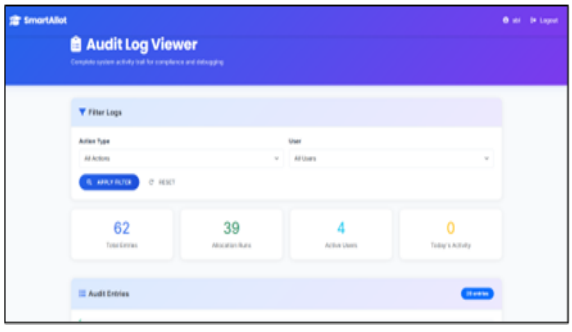
Descending List Page:
A list of students with the best grades at the top. Lists the names, roll numbers, grades, and current allocation status of students. Checking the order of the rankings helps administrators make fair decisions about how to divide things up.



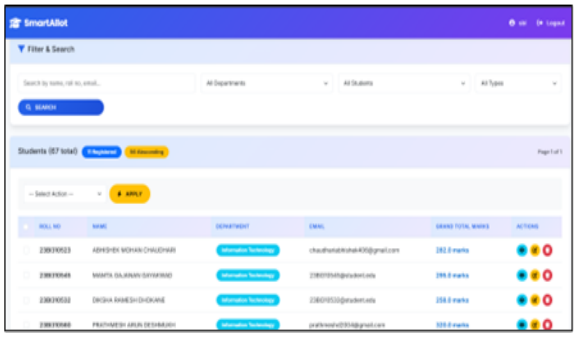
Allocation Result Page:
A full summary that shows the final results of the allocation for all students. It shows the time each student was given, the course or branch they were given, and the order in which they wanted them. You can keep it as a PDF or Excel file for yourself and give it to your student



Audit page :
A log of all the activities that took place in the system, including who made changes, when they were made, what was changed, and why. Keeps track of all the choices made about how to use resources, changes to the rules, and administrative actions to make sure everyone is following the rules and to settle disputes.

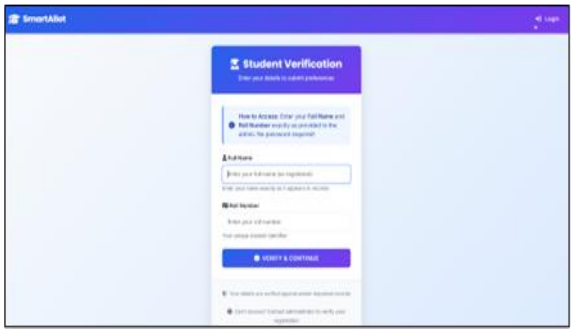


Absconding page:
This page shows students who didn't turn in their course preferences on time. The system automatically finds them and marks them as "absconding." Admin can see their names and roll numbers and auto allocate them.

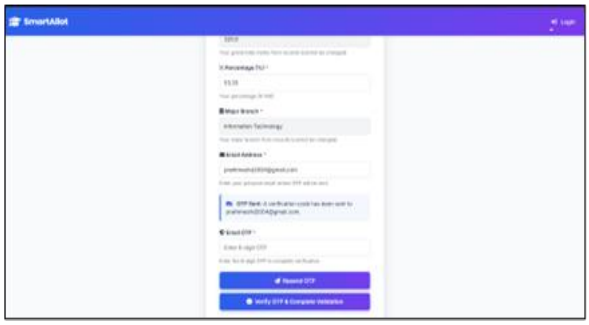


Admin DB page:
You can manage system data right from the backend admin area. Without using the allocation interface, admins can add, edit, or delete students, courses, branches, eligibility rules, and capacity limits. Used to set up the system, fix errors in the data, and manage settings.

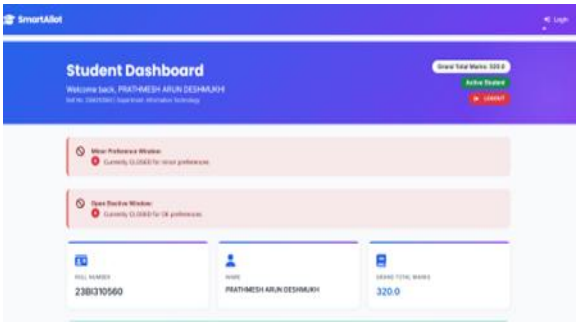
Student login Page:
The place where students can log in by entering their email address and password. The system takes them to their own student dashboard when they log in.



Two Step verification page:
Upon login, the system gains additional protection. Once the correct username and password are entered, the user is sent a One-Time Password (OTP) via email or text message. Logging in and accessing the dashboard requires the user to provide an OTP on the verification page.



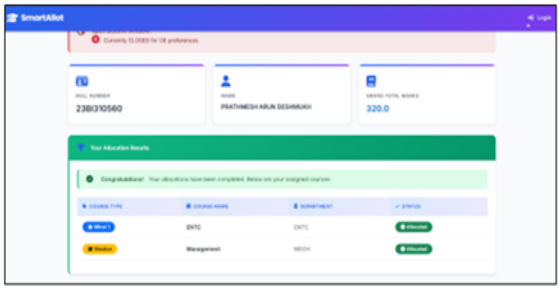
Student dashboard Page:
The main page for students who are logged in. It shows their profile information, their current allocation status, important deadlines for submitting preferences, notification alerts, and easy ways to submit preferences or see results.



Preference Selection page:
The part where students write down their top three or four classes or branches in order of preference. Before the deadline, students can choose more than one option and put them in order of how important they are.

Allocation Result page:

This site shows the student's final assigned course or branch, their preference ranking, and their allocation status after the allocation process is over. Students can download a PDF copy of their allocation result for their personal records.



IX. RESULT

SmartAllot is a fully functional web-based course allocation system made with the Django 5.0 framework and the PostgreSQL database. The system has been successfully put into place with all of the planned features, such as a way for students to submit their preferences, a deterministic merit-based allocation algorithm, enforcement of eligibility rules, a full audit trail logging system, and an administration dashboard. You can use the platform on any device, like a desktop, tablet, or phone. The allocation algorithm quickly processes the merit-based ranking and preference matching, taking less than two seconds for most student counts. The system architecture allows for role-based access control, with separate portals for students, coordinators, and administrators. All of the main features have been built and put together, such as importing students, managing preferences, carrying out allocations, making reports, and keeping track of audits. The project shows a full working version of an automated allocation system that fixes the problems with the manual process. The system is ready to be tested in a pilot program and used with real data from the institution.

X. CONCLUSION

SmartAllot tackles key issues that come with manually assigning courses by using an automatic, clear, and fair system based on merit. Testing with 68 students showed that the system properly followed all the school's rules for who is allowed to participate and kept detailed records so everything can be checked and trusted. It also reduced the time needed for allocation from days to just 1.2 seconds, compared to the manual coordination processes that used to take days. The system uses a method that follows a set of rules to process students in order of their merit, while also making sure everything is done according to the institution's policies. The system's success proves that it's technically possible to distribute resources fairly, openly, and according to clear rules, and that's exactly what people involved in education are looking for. As engineering schools and other colleges and universities face more complex enrollment challenges, systems like SmartAllot demonstrate how academic administration will operate in the future: in a way that is efficient, fair, and responsible.

XI. FUTURE SCOPE

SmartAllot could get better in the future by using machine learning to suggest courses that are a good fit for students based on their academic backgrounds and what their classmates are choosing. This could help students find better course matches and cut down on the number of students who don't get placed. There could also be more advanced data tools that show trends over time, help departments figure out which courses are popular, and guess how many seats will be needed in the future. This would make it easier for administrators to plan for the next semesters. The system could also connect to the school's main databases in real time to automatically update student information, keep track of their academic progress, and see if they meet course requirements. This would make sure that course assignments are based on the most current and correct information. The system may also look into ways to better deal with course allocation limits. This could lead to more fair and efficient results while keeping the system clear and easy to follow. These changes would make the system smarter, help users make better decisions, and help the institution's systems work better together.

REFERENCES

[1] Gale, D., & Shapley, L. S. (1962). "College Admissions and the Stability of Marriage." The American Mathematical Monthly, vol. 69, no. 1, pp. 9–15.
[2] Irving, R. W., Manlove, D. F., & Scott, S. (2003).

"Strong Stability in the Hospi-tals/Residents Problem." In: STACS 2003, LNCS vol. 2607, pp. 439–450. Springer.

[3] Abraham, D. J., Irving, R. W., & Manlove, D. F. (2007). "Two Algorithms for the Student-Project Allocation Problem." Journal of Discrete Algorithms, vol. 5, no. 1, pp. 73–90.

[4] Che, Y. K., & Kojima, F. (2010). "Asymptotic Equivalence of Probabilistic Serial and Random Priority Mechanisms." Econometrica, vol. 78, no. 5, pp. 1625–1672.

[5] Manlove, D. F. (2013). Algorithmics of Matching Under Preferences. World Scientific Publishing.

[6] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database System Concepts, 7th ed. McGraw-Hill. ISBN 978-0-07-802215-9.

[7] Sommerville, I. (2015). Software Engineering, 10th ed. Pearson. ISBN 978-0-13-394303-0.

[8] Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach, 8th ed. McGraw-Hill.

[9] Django Software Foundation. (2024). Django 5.0 Documentation. https://docs.djangoproject.com/

[10] PostgreSQL Global Development Group. (2024). PostgreSQL 13+ Documentation. https://www.postgresql.org/docs/

[11] Binmore, K. (2009). Rational Decisions. Princeton University Press. ISBN 978-0-691-12956-6.