Automated Exam Room Allotment Using Cloud- Serverless Computing

Prof. Indumathi S[1], Prof Pavan Mulgund[2], Prof Shilpa Prabhu Patil[3] and Dr Sampath Kumar Y R [4]

[1] *Assistant Professor, Department of Artificial Intelligence & Machine Learning, BMS Institute of Technology & Management, Bengaluru, Karnataka, India, indusrinivasa@gmail.com*

[2]*Assistant Professor, Department of Artificial Intelligence & Machine Learning, BMS Institute of Technology & Management, Bengaluru, Karnataka, India, pavan.mulgund24@gmail.com*

[3]*Assistant Professor, Department of Artificial Intelligence & Machine Learning, BMS Institute of Technology & Management, Bengaluru, Karnataka, India, shilpa.prapatil@gmail.com*

[4]*Assistant Professor, Department of Artificial Intelligence & Machine Learning, BMS Institute of Technology & Management, Bengaluru, Karnataka, India, yrsam008@gmail.com*

Abstract : Effective room allocation for examinations is an important administrative problem in educational institutions. Common methods, both manual and automated, do not succeed in avoiding clustering of students enrolled in the same course, hence there is a possibility of cheating. In this paper, we present an adaptive multiple constraint-based approach for room allocation implemented via a serverless cloud-based architecture hosted on Amazon Web Services (AWS). The approach considers subject-based distribution, department balancing, and an adaptive soft constraint based on which the number of students from the same subject per room is limited, along with automatic fallback to other departments where available. Moreover, the proposed method takes dynamic input parameters such as the room size and number of rooms during the execution process. The proposed algorithm runs on AWS Lambda, with data stored in Amazon S3 and request handling via the API Gateway. The experimental analysis shows that our approach ensures balanced room allocation without clustering of the same subjects in the same room without any constraints violations regardless of the dataset size.

Keywords: Serverless Computing; AWS Lambda; Cloud Computing; Exam Room Allotment; Automation

I. Introduction

While examinations conducted by educational institutions are a regular exercise, they are still considered one of the more difficult administrative responsibilities. This is even more so in universities and engineering colleges, where there is increased strength and diversity of departments. One of these tasks, room allocation, impacts the actual examination process, and poor room allocation can cause small blunders that can be disastrous on the day of the examination. The size of the room, the fair representation of the different departments, and fairness have been discussed in the research works on examination timetabling and scheduling [1] to [5].

Despite research progress on timetabling, many institutions still assign rooms by hand or using spreadsheets. Both approaches are feasible, but as student numbers increase they do not scale well and become more difficult to control and more prone to errors. Prior timetabling research has stressed the importance of

While there are similar scalability and coordination challenges for semi-automated systems [2], [3], administrative bottlenecks (especially at peak examination times) are usually caused by infrastructure and workflow deficits, rather than algorithmic limitations.

More recently, there have been automatic approaches to this problem, utilizing a constraint-based problem formulation or heuristics [4], [5]. With the emergence of cloud computing, opportunities for deployment have expanded even further. Many of the cloud computing characteristics, such as elasticity, on-demand resource provisioning, outlined in the NIST definition apply to academic systems (see [7]). However, practical implementations often rely on virtual machines or existing server-hosted models [6]. Such methods work but require continuing administration and maintenance, with usually-provisioned resources used even when idle, which is less than ideal in an academic context with workloads varying over the academic year.

Serverless computing is an architectural style where servers are not provisioned, but running application logic is fully managed and triggered by various events. The cloud provider is responsible for and manages infrastructure, scalability and fault tolerance [8], [12]. Survey articles of serverless computing platforms have reported benefits of serverless architectures including

ease of operation and low cost of bursty workloads [9] [11] [13].

The suitability of Function-as-a-Service for such an application is further supported by empirical studies of Function-as-a-Service platforms [8], [14], as well as the observed peak demand during examination periods and relatively low demand otherwise.

Despite serverless computing being a much-discussed topic in recent years, there is little research done on serverless platforms in the context of the exam room allotment problem. However, most of the work has either focused on algorithms to optimize scheduling problems [1] to [5] or comparing serverless platforms [8] to [13]. There is little discussion on how to design a serverless event-driven architecture to solve administrative problems like room allotment, though serverless components in educational cloud systems have started to be explored [15]. However, no clear architectural blueprint for a serverless architecture for room allotment has been laid out.

To address this shortfall, we present a serverless cloud-based application for automatically assigning examination rooms using Amazon Web Services (AWS). Our proposed solution builds an event-driven allocation pipeline using Amazon's Simple Storage Service (S3), API Gateway, and AWS Lambda. However, since we also consider room capacities and department distribution in our capacity-aware model, we focus on architectural feasibility, scalability, and institutional applicability to yield a deployable campus-level solution. In this context, we leverage the elasticity and pay-per-use properties of serverless platforms [8] to [13] to achieve compliance to operational constraints of academic institutions.

The contributions of this work include:

- Implementation of a room allotment framework entirely on a serverless architecture.
- Development of a constraint-based allocation mechanism that supports balanced distribution.
- Evaluation of scalability behavior under varying dataset sizes.
- Analysis of operational suitability for institutions with fluctuating workloads.

The remainder of the paper is structured as follows. Section II discusses related literature in more detail. Section III presents the architectural design. Section IV describes the allocation process. Section V outlines experimental observations. Section VI concludes the study

A. The remaining sections of this paper are as follows. Section II reviews the related work on exam scheduling and serverless applications. Section III describes the proposed system architecture. Section IV presents implementation details. Section V discusses results and performance evaluation. Finally, Section VI concludes the paper and outlines directions for future work

II. RELATED WORK

Examination timetabling and room assignment have been studied for many decades, with the early work focusing on automatic timetabling and constraint satisfaction problems (CSP). Early review papers for timetabling are provided by Schaerf [1] and Burke and Petrovic [2]. These reviews explore the difficulty in enforcing conflicting constraints. Numerous attempts were made to extend heuristics and metaheuristics, such as tabu search, to model the exam scheduling problem and other optimization problems [4][5]. These efforts were influential for the use of computational methods for scheduling constraints.

Subsequent work has surveyed a wider variety of school and university timetabling problems. Pillay [3] surveyed school timetabling research, with an emphasis on scalability. These studies improved the algorithmic approaches, but were mainly focused on timetable generation rather than the architectural implementation of room allotment systems.

Due to the emergence of the cloud, research on the deployment of academic management systems to the cloud has been conducted. [6] studied the scheduling and optimization of the cloud and briefly introduced elastic resource management. NIST's cloud computing model introduced on-demand self-service, resource pooling, and measured service as properties of cloud computing [7]. Their documents provided some clarity about the migration of institutional systems to the cloud, although much of what passed for cloud computing in those years was just virtual machines in server farms that still needed maintenance and configuration. Another architectural pattern emerged for serverless computing. [8] discussed serverless architecture and its performance. [9] described the evolution of architectural vision for event-driven programming in the cloud.

[10] and [11] analyze trends, advantages, and open issues in serverless computing. Leitner et al. [12] conduct a multivocal literature review and also note the main benefits to be cost efficiency and operational simplicity. Subsequent work focused on the reliability and performance of serverless environments [13] and the more implementation specific aspects of Function-as-a-Service runtimes [14].

Cloud examination systems are proposed as scalable and accessible modes of education assessment. Malakar and [15] proposed a cloud examination framework using microservices and serverless architecture. They proposed cloud-native educational systems but did not provide a wide-ranging review of a fully server-less event-driven room allocation process.

Previous work in this area has mainly focused on (i) developing algorithms for timetabling and constraint satisfaction [1], [5] or (ii) developing architectures for cloud and serverless computing systems [6], [14].

While some early work has been done in the area of cloud computing systems for education [15] there has not been a focused combination of constraint aware logic for room assignment with a fully serverless event-driven architecture and institutional administrative workflows.

To highlight the evolution, Table I compares traditional, cloud-hosted, and serverless approaches for exam management systems.

Table I: Comparison of Exam Management System Approaches

Feature	Traditional Systems	Cloud-Hosted Systems	Serverless Systems (Proposed)
Deployment	On-premise, manual setup	Hosted on VMs or servers	Fully managed on cloud (no servers)
Scalability	Limited, hardware-dependent	Moderate (requires scaling servers)	Automatic, event-driven Scaling
Cost Model	High upfront + maintenance	Pay for reserved instances	Pay-per-use (only when functions run)
Performance during Peak Load	Often bottlenecks	Better but needs tuning	Auto-scaling, seamless handling
Maintenance	Requires IT staff	Needs updates & patching	Minimal (managed by provider)

III. System Architecture

The serverless architecture of the system allows automatic room allocation without the need for continuous back-end servers, as the workflow is only executed when required and specific conditions are met. This allows the system to go idle when a dataset file is not submitted for examination by the user.

The architecture consists of three layers: the user interface, the processing layer, and the storage layer. These are loosely coupled to each other through event-driven messaging.

A. User Interaction and Request Handling

The user interface is a static web application built using standard HTML and JavaScript and hosted using Amazon S3 as a static website. Because the user interface does not need to be rendered by the back-end, a web server does not need to be maintained.

The administrator uses this web interface to request a secure upload link, upload the dataset of students, and download the resulting allocation.

Gateway. The API endpoint triggers a serverless function to generate a temporary pre-signed URL. It can upload directly to the storage bucket via a temporary upload link thus not requiring system credentials. This makes it easier to grant access while keeping the overall code/interface simple.

B. Allocation Processing Mechanism

This allocation logic is written as an AWS Lambda function, that is invoked everytime a new file with dataset is uploaded to the input storage bucket.

When function is invoked, it reads the uploaded file, assign students to rooms and write the output file showing student-room pairing. This feature has scheduling disabled and therefore does not spend resources when it is idle, since this function is only called after the file was uploaded.

C. Data Storage and Workflow Sequence

Input and output files are stored in Amazon S3, in separate buckets for inputs uploaded by users and for allocations generated by the system. Because the inputs and outputs are stored as structured text files, object storage is sufficient, and no database layer is required.

The operation is simple: the administrator touches the web interface and requests an upload link. Next, the data set is uploaded. The upload process calls the function that processes the upload. Once completed, the successfully allocated file will be available at the specified location.

This structure reduces reliance on infrastructure and maintenance, while also aligning with the academic workloads concentrated around examination periods.

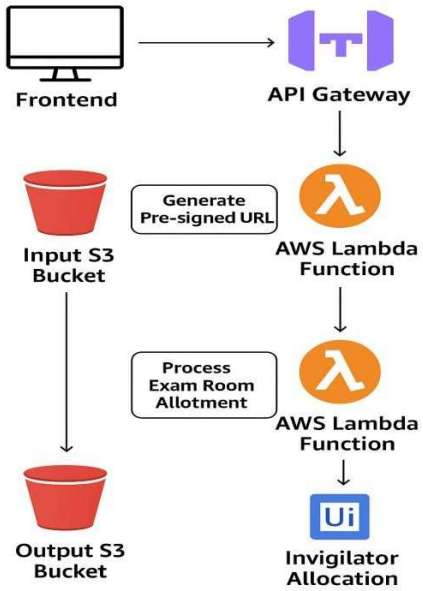

Figure 1. System Architecture of the Serverless Exam Room Allotment System

IV. Methodology

The proposed methodology focuses on designing an adaptive and constraint-aware room allocation strategy that ensures fair distribution of students while maintaining examination integrity. Unlike traditional cyclic or static allocation approaches, the proposed method incorporates subject-aware distribution, soft constraint enforcement with fallback relaxation, and dynamic infrastructure inputs. The objective is to minimize subject clustering while maximizing room utilization under practical constraints.

A. Problem Definition

Let:

- $S = \{s_1, s_2, \ldots, s_n\}$denote the set of students
- $R = \{r_1, r_2, \ldots, r_m\}$denote the set of available rooms

Each student $s_i \in S$is associated with:

- a subject $sub(s_i)$
- a department $dept(s_i)$

Each room $r_j \in R$has a maximum capacity C.

The objective is to determine a mapping:

$$f: S \rightarrow R$$

such that each student is assigned to exactly one room while satisfying allocation constraints.

B. Constraints Formulation

1. Room Capacity Constraint (Hard Constraint)

For each room $r_j \in R$:

$$\sum_{s_i \in S} \mathbb{I}(f(s_i) = r_j) \leq C$$

where $\mathbb{I}(\cdot)$is the indicator function.

2. Subject Constraint (Soft Constraint)

To reduce clustering of students appearing for the same examination:

$$\sum_{s_i \in S} \mathbb{I}(f(s_i) = r_j \wedge sub(s_i) = k) \leq \frac{C}{2}, \forall k$$

This constraint is treated as a **soft constraint** and may be relaxed when no alternative subjects are available.

3. Department Constraint

To maintain fairness:

$$\sum_{s_i \in S} \mathbb{I}(f(s_i) = r_j \wedge dept(s_i) = d) \leq D$$

where $D$is the maximum allowed students from a department per room.

4. Feasibility Constraint

$$|S| \leq |R| \times C$$

If this condition is violated, allocation is not feasible.

C. Objective Function

The allocation aims to reduce the number of students from the same subject sitting in a single room. This is achieved by minimizing the number of same-subject students assigned to each room:

$$\min \sum_{r_j \in R} \sum_{k} \left(\text{count}_{r_j}(k)\right)^2$$

where $\text{count}_{r_j}(k)$is the number of students of subject $k$in room r_j.

At the same time, room space should be used efficiently:

$$\max \sum_{r_j \in R} \frac{|r_j|}{C}$$

where $|r_j|$is the number of students in room r_j, and $C$is the room capacity.

D. Allocation Strategy

The allocation follows the **Adaptive Multi-Constraint Room Allocation Algorithm (AMCRAA)**.

Students are grouped based on subject and department. Rooms are filled one by one.

For each room:

- Students are selected to reduce repetition of the same subject
- Department balance is maintained
- A maximum of $C/2$students from the same subject is allowed

If no other subject is available:

- The subject limit is relaxed (fallback)

The process continues until all students are assigned.

E. Algorithm Description

Input:

Student set S

Rooms R

Capacity C

1. If |S| > |R| × C:
   return "Allocation Not Feasible"

2. Group students by (subject, department)

3. For each room r_j in R:
   Initialize subject_count and dept_count

   While |r_j| < C and S not empty:

      Select student s_i such that:
         subject_count[sub(s_i)] < C/2
         and department constraint satisfied

      If such student exists:
         Assign s_i to r_j
      Else:
         Assign any remaining student (fallback)

      Update counts

4. Return allocation mapping f

F. Computational Complexity

Let $n = | S |$:

- Grouping students: $O(n)$
- Allocation process: $O(n)$

Overall complexity:

$$O(n)$$

This linear complexity ensures efficient execution within serverless environments such as AWS Lambda.

G. Design Rationale

The proposed method focuses on a simple and practical approach instead of complex optimization techniques. By considering subject constraints, the system helps reduce the number of students writing the same exam in one room. The use of a soft constraint with fallback ensures that allocation is completed even when ideal conditions are not possible. In addition, allowing inputs such as room count and capacity makes the system flexible for different examination setups.

V. Experimental Setup

A. Deployment Environment

The system was deployed using:

- AWS Lambda (Python runtime)
- Amazon S3 (Input and Output Buckets)
- API Gateway (REST interface)

The experiments were conducted under AWS Free Tier configuration.

B. Dataset Configuration

Synthetic datasets were generated with varying student sizes:

Dataset	Number of Students	Number of Departments
D1	100	3
D2	250	5
D3	500	8

Room capacity was fixed at 40 students.
Maximum departmental limit per room was set to 20.

C. Evaluation Metrics

The following performance metrics were measured:

1. Allocation Execution Time (seconds)
2. Number of Rooms Utilized
3. Constraint Violations (should be zero)
4. Estimated Execution Cost (AWS billing model)

VI. **Results and Discussion**

The proposed system was successfully deployed and tested on AWS cloud. Figures 3–6 illustrate key implementation details and outputs of the automated exam room allocation system."

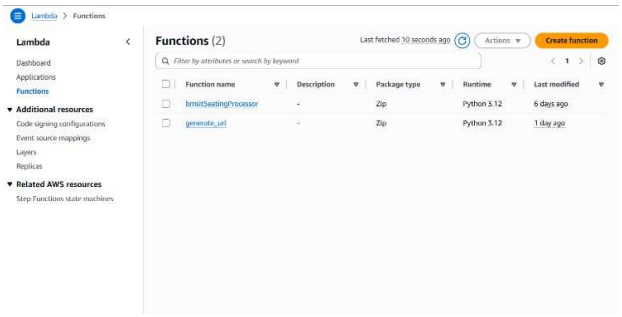

Figure 2: AWS Lambda console showing two functions (generate_url andseatingProcessor).

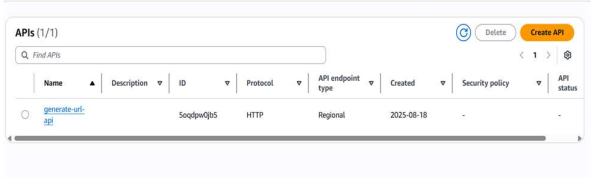

Figure 3: API Gateway endp oint /generate_url configured.

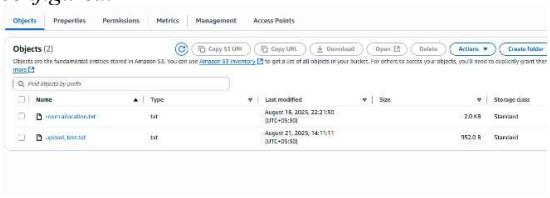

Figure 4: S3 input buckets with uploaded input file.

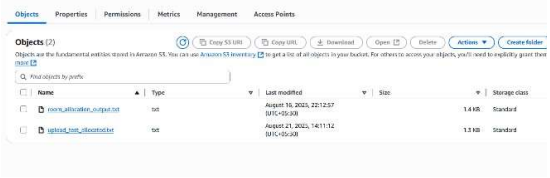

Figure 5: S3 output buckets with generated output file.

Figure 2 shows the AWS Lambda console, here two functions are deployed. The *generate_url* function and*seatingProcessor* function. The *generate_url* function is responsible for creating pre-signed URLs to securely upload input text files, and the *seatingProcessor* function runs the allocation algorithm on input file upload.

Figure 3 presents the API Gateway configuration with the */generate_url* endpoint. This endpoint interacts with the Lambda function to generate the upload link. this link is used by the administrator to upload the input file.

Figure 4 shows the input S3 bucket,the input text file will be residing here.once file is been uploaded, This action triggers the *seatingProcessor* Lambda function automatically through an S3 event notification.

Figure 5 shows the output S3 bucket, which contains the final exam room allotment file that is the output file. The generated file includes student USN, department, and assigned room number.

Performance Evaluation

Datasets of varying dimensions were used to analyze the system performance. Room maximum was 40 students and departmental maximum/room was 20 students.

Students	Execution Time (seconds)	Rooms Used	Violations
100	0.41	3	0
250	0.89	7	0
500	1.76	13	0

When the number of students increased, execution time also grew proportionately in a linear manner. There were no violations in any dataset. Dormitory occupancy was kept reasonable and within established limits.

Given that the system is developed to operate in a serverless environment, no manual scaling was done. Execution is on the event-driven basis, which minimizes wastage of resource when idle. The cost was very low, according to the AWS Free Tier thresholds.

In summary, our results indicate that the introduced framework is robust, expansible due to its scalability with dataset size and provides support for automatic exam room allocation without cross-tenant infrastructure overhead.

VII. Conclusion

This Paper has demonstrated an effective cloud-based serverless approach for allocating examinations in an automated manner. The proposed approach is much better than the conventional approach because it takes into consideration the distribution of subjects, which will minimize the tendency of students to be clustered into rooms for the same examination. Moreover, there is an adaptive constraint, and also, the system falls back if necessary so as to assign every stu-dent to a room. Also, since one can input values such as room size, the system is able to adapt easily. As observed from the simulation results, the algorithm has been able to work effectively and efficiently with the increasing number of stu-dents, but without violating any of the constraints set.

VIII. Future Enhancements

The current system can be further extended by the inclusion of such features as timetable integration. In future, more sophisticated optimization algorithms could be implemented to improve the effectiveness of allocation. The ability to coordinate operations across multiple campuses could also be considered, along with the possibility of having real-time monitoring capabilities within the system**.**

environments.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

[1] S. Ceschia, L. Di Gaspero, and A. Schaerf, "Educational timetabling: problems, benchmarks, and state-of-the-art results," *Computers & Operations Research (preprint / arXiv)*, Jan. 2022
[2] E. Siew, G. Kendall, and colleagues, "A survey of solution methodologies for exam timetabling," *arXiv / technical report*, 2024.
[3] A. Muklason, "Generic university examination timetabling system with hill-climbing and hybrid heuristics," *Procedia / ScienceDirect*, 2024

[4] C. W. Fong, "University examination timetabling using a hybrid black-box heuristic," *Journal of Informatics and Innovation in Virtualization (JOIV)*, 2022.

[5] E. S. K. Siew, S. N. Sze, S. L. Goh, M. Hossin, and K. L. Chiew, "A survey of solution methodologies for exam timetabling," *Technical Report / arXiv*, 2024.

[6] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, "Survey on serverless computing," *Journal of Cloud Computing*, vol. 10, Art. no. 39, 2021

[7] Z. Li et al., "The Serverless Computing Survey: A Technical Primer for Research and Practice," *ACM / Communications / survey (2022)*.

[8] J. Wen, et al., "Rise of the planet of serverless computing: A systematic literature review," *ACM / TOSEM (2023)*.

[9] S. Mohanty et al., "FAAS: Optimizing loop-based applications for serverless," (system / arXiv / preprint), 2024 — useful for function-level optimizations and cold-start / packing strategies.

[10] H. Ship, E. Shindin, C. Wang, et al., "Optimizing simultaneous autoscaling for serverless cloud computing," *arXiv (2023)* — good for autoscaling/resource-allocation theory in serverless platforms

[11] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, "Survey on serverless computing," Journal of Cloud Computing, vol. 10, Art. no. 39, 2021.

[12] Z. K. Ozturk, "Exam scheduling under pandemic conditions: A multi-objective MIP and heuristic approach," *ScienceDirect / Elsevier*, 2024

[13] B. Bassimir, "On the computation of robust examination timetables," *Journal of Scheduling*, 2024.

[14] C. Spillner, "Snafu: Function-as-a-Service (FaaS) runtime design and implementation," in Proc. IEEE/ACM SEAMS, 2017, pp. 94–100.

[15] A. Lemos, P. T. Monteiro, and I. Lynce, "University course timetabling with MaxSAT," *Journal of Scheduling*, vol. 25, no. 3, 2022.