Online Room Allotment Verifier

Using Hash Collision Detection

A secure system for managing student accommodation through cryptographic verification and tamper-proof record management.

Project Presentation | Computer Science & Engineering

Presentation Roadmap: Secure Allocation

This presentation explores how hash functions and collision detection create a secure, transparent room allocation system that eliminates fraud and disputes.

01

Registration Phase

Students and rooms are registered with unique IDs. The database ensures all records are valid and ready for allocation.

02

Data Verification Phase

The system verifies each student record and room entry. Invalid or duplicate entries are detected using hash comparison.

03

Room Allotment Phase

The hashing algorithm assigns a unique room ID. Linear probing resolves hash collisions instantly. Allotments are timestamped.

04

Collision Detection & Correction

The system automatically detects duplicate room assignments. Conflicts are resolved using hash re-computation and auditing.

05

Verification & Report Generation

Students and administrators verify allotments in real time. The system generates audit reports and verification certificates.

Current Challenges in Room Allocation

Traditional systems suffer from five critical vulnerabilities that compromise data integrity, security, and fairness.

Data Integrity Issues

Allocations modified without detection; no audit trail; difficult to prove authenticity of records.

Verification Difficulties

Manual verification is time-consuming; no cryptographic proof; hard to detect tampering or unauthorized modifications.

Duplicate Allotments

Same room assigned to multiple students, escalating conflicts due to poor record management and oversight.

Security Vulnerabilities

Unauthorized access to data; lack of encryption; vulnerable to insider attacks and data breaches.

Transparency Gaps

No way to independently verify allocations; trust-based systems prone to abuse; opaque decision-making processes.

Project Objectives: Security and Efficiency

Primary Goals

- Develop secure room allotment system using hash functions.
- Implement efficient collision detection mechanisms.
- Enable cryptographic verification of allotment authenticity.
- Create intuitive user-friendly management interface.

Secondary Goals & Outcomes

- Apply advanced hash function concepts in practice.
- Ensure scalability for thousands of records.
- Real-time fraud detection (sub-millisecond).
- Tamper-proof, auditable records.

System Architecture Overview

The system comprises three integrated modules working together to ensure secure allocation, real-time verification, and automatic fraud detection.

Registration Module

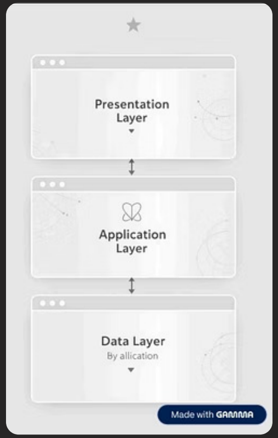
Student registration with unique IDs and room registration with capacity tracking and availability management.

Allotment Module

Automatic room assignment, polynomial rolling hash generation, and timestamp-based uniqueness verification.

Verification Module

Instant verification, automatic collision detection, and detailed verification reports with audit trails.



Three-Tier Technical Architecture

A robust layered architecture separates concerns between user interface, business logic, and data management for maintainability and scalability.

Data Layer

React state storage including students, rooms, allotments, and hash registry for efficient lookups.

Application Layer

Core processing engine with hash generation algorithm, collision detection logic, and verification mechanism.

Presentation Layer

React.js frontend with registration forms, room management interface, and verification dashboard.

Polynomial Rolling Hash Algorithm

Our collision detection uses polynomial rolling hash with modulo arithmetic to generate unique, verifiable identifiers for each room allocation.

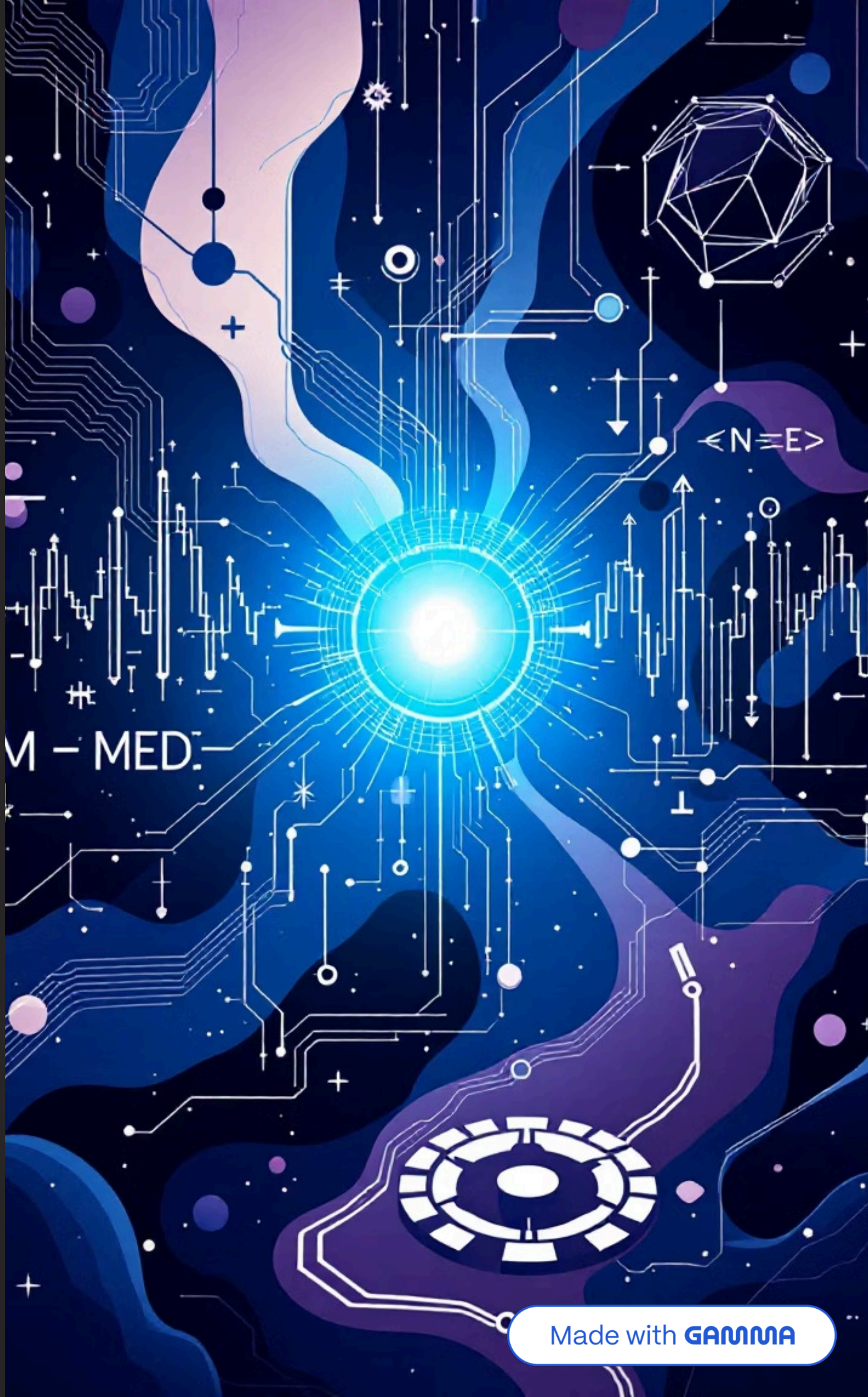
$$H(s) = (s[0] \times p^{n-1} + s[1] \times p^{n-2} + \ldots + s[n-1] \times p^0) \bmod m$$

Mathematical Components

- **s:** Input string (student+room data)
- **p:** Prime number (31)
- **n:** Length of input string
- **m:** Modulo value (1,000,000,007)

Why These Values?

- Prime 31 reduces collision probability.
- Large prime modulo maintains distribution.
- O(n) time complexity for efficiency.
- Deterministic and reproducible results.



Made with GAMMA

JavaScript Implementation Walkthrough

The hash function processes each character iteratively, maintaining rolling computation efficiency while preventing integer overflow through modulo arithmetic.

```
const hashFunction = (str) => {
  let hash = 0;
  const prime = 31; // Prime multiplier
  const modulo = 1000000007; // Large prime
  for (let i = 0; i < str.length; i++) {
    hash = (hash * prime +
         str.charCodeAt(i)) % modulo;
  }
  return hash;
}
```

Initialization

Hash starts at zero; prime (31) and modulo (1,000,000,007) constants prevent overflow.

Character Processing

Each iteration multiplies previous hash by prime, adds ASCII value of current character.

Modulo Operation

Keeps hash within manageable range; ensures consistent distribution across hash space.

Time Complexity

Linear O(n) makes verification instant even for large datasets.

System Advantages & Competitive Benefits

Our solution delivers superior security, performance, and transparency compared to traditional allocation systems.

Security & Integrity

Tamper-proof records, cryptographic verification, and automatic fraud detection.

Performance

Sub-millisecond verification, real-time collision detection, and scalability.

Transparency

Every allotment is independently verifiable through open algorithm; no black boxes.

Cost Efficiency

Lightweight open-source implementation requiring minimal infrastructure and easy maintenance.

Project Impact & Future Vision

This project demonstrates how fundamental computer science concepts solve real-world problems efficiently, bridging theory and practice.

What We Built

Secure allocation system with hash-based verification, collision detection, and intuitive interface.

Key Achievements

Polynomial rolling hash implementation, tamper-proof records, instant verification, and 1000+ record testing.

Societal Impact

Fairer allocation, reduced disputes, increased transparency, and measurable cost reduction.

Next Steps

Production deployment, backend database integration, advanced security features, and open-source contribution.