Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Ayojanix: Design and Testing of a QR-Based Event Management System for Colleges

Dr. Shrikant P. Akarte , Apurva S. Kadu , Arya A. Raut , Vedant R. Himte, Tejas D. Nikose
Prof. Ram Meghe Institute of Technology and Research Badnera - Amravati

***Abstract*—Colleges often manage events with separate tools such as Google Forms, paper attendance sheets, spreadsheets and WhatsApp groups. This causes long queues at entry, proxy attendance, food waste and certificates that are easy to fake. This paper presents Ayojanix, a web-based event management system built with React, Node.js, Express and a PostgreSQL database accessed through Prisma ORM. Students register online and receive a unique QR code pass. Volunteers scan the pass at the gate, and the server rejects invalid, wrong-event and already-used passes to stop proxy entry. The system uses role-based access control (RBAC) with JWT authentication for four roles: Admin, Organizer, Volunteer and Participant. A rule-based decision support module predicts food demand, analyses feedback sentiment and suggests volunteers, budget and venue. Certificates include a QR code for public verification. Functional testing confirmed that the system correctly accepts valid passes, rejects duplicate scans, supports manual pass entry, calculates food demand as designed and verifies certificates.**

***Keywords*—Event Management System, QR Code Attendance, Role-Based Access Control, Rule-Based Decision Support, Food Demand Prediction, Sentiment Analysis.**

I. INTRODUCTION

Colleges organize many events every year, such as hackathons, seminars, workshops, conferences and cultural fests. Each event needs registration, attendance, food, volunteers, money records and certificates. Most colleges still manage these tasks with separate tools. Registrations come through Google Forms, attendance is marked on paper, accounts are kept in spreadsheets, and messages go out on WhatsApp groups.

These tools do not share data, which creates four common problems:

- **Long queues at the entry**, because volunteers search for names on paper lists.
- **Proxy attendance**, because one student can sign for another, or one pass can be shared as a screenshot.
- **Food waste**, because meals are ordered for every registered student even though many do not attend.
- **Fake certificates**, because PDF or image certificates are easy to edit and hard to verify.

In our earlier work [1], we reviewed existing event management systems and found that none of them combines all the features colleges need in one platform. In this paper, we present the design, implementation and testing of that platform, called **Ayojanix**.

Ayojanix is a web application with a three-tier architecture: a React frontend, a Node.js and Express backend, and a relational database accessed through Prisma ORM. It supports four user roles (Admin, Organizer, Volunteer and Participant) through role-based access control (RBAC).

The main contributions of this paper are:

1. A QR code attendance system that checks every scan on the server and rejects invalid passes, passes for other events and already-used passes, which prevents proxy entry.
2. A rule-based decision support module that predicts food demand, analyses feedback sentiment, and suggests the number of volunteers, the budget split and the venue for an event.
3. Certificates with a QR code that anyone can scan to check whether the certificate is genuine or revoked.
4. Functional testing of the main features, with a plan for evaluation on live college events.

II. RELATED WORK

Shah et al. [2] built a web-based event booking system using React and Express, where customers book venues and services and chat directly with the event provider. It was tested by three users. However, it focuses on private events such as weddings, and it has no attendance tracking or certificates.

Kulkarni et al. [3] developed a web tool for educational institutions with modules for registration, login, event creation, event registration, reports and feedback. It replaces paper-based records, but attendance and feedback are handled without any automatic checking or analysis.

Thombare et al. [4] proposed a college event management system with event creation, student registration and automated notifications. The paper describes a planned system and methodology but does not report an implementation or test results.

Mayan-ao et al. [5] developed an events management system using Laravel and MySQL, with booking, budget tracking, vendor coordination and SMS reminders. Its usability was rated “Good” on the System Usability Scale. However, it is designed for an event styling business, not for colleges, and it has no QR attendance.

Sakeri et al. [6] developed SmartEMS for small businesses, with separate organizer, participant and admin pages, event approval by the admin and e-certificate claims. However, attendance is collected through an external form link, and certificates cannot be verified by a third party.

Table I compares these systems with Ayojanix.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

TABLE I. FEATURE COMPARISON

Feature	[2]	[3]	[4]	[5]	[6]	Ayojanix
Online registration	✓	✓	✓	✓	✓	✓
QR attendance with check-in/check-out	✗	✗	✗	✗	✗	✓
Proxy / duplicate entry prevention	✗	✗	✗	✗	✗	✓
Finance and invoices	✗	✗	✗	✓	Partial	✓
Volunteer duty management	✗	✗	✗	✗	✗	✓
Food demand prediction	✗	✗	✗	✗	✗	✓
Feedback sentiment analysis	✗	✗	✗	✗	✗	✓
Publicly verifiable certificates	✗	✗	✗	✗	✗	✓

From this comparison, no existing system combines QR attendance, proxy prevention, food prediction, feedback analysis and verifiable certificates for college events. Ayojanix is designed to fill this gap.

III. SYSTEM DESIGN

A. System Architecture

Ayojanix uses a three-tier client-server architecture, as shown in Fig. 1.

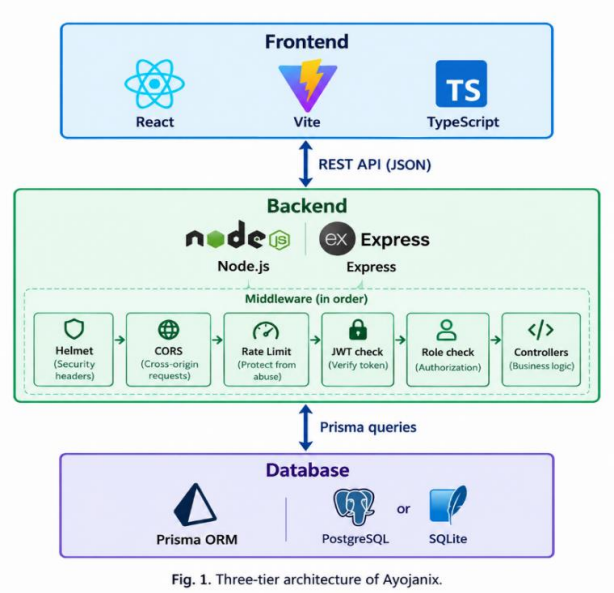


Fig. 1. Three-tier architecture of Ayojanix.

- **Frontend:** a single-page web app built with React and TypeScript. It runs in any browser, including mobile phones, so volunteers do not need to install an app. The QR scanner uses the phone camera through the html5-qrcode library.
- **Backend:** a REST API built with Node.js and Express. Every request passes through security steps in a fixed order: secure HTTP headers (Helmet), allowed-origin check (CORS), a rate limit of 1,000 requests per 15 minutes per IP address, JWT token check, and role check. Only then does the request reach the main logic.
- **Database:** a relational database accessed through Prisma ORM. SQLite is used during development and PostgreSQL in production. Prisma uses parameterized queries, which protects the system from SQL injection.

We built the system as one server (a modular monolith) instead of separate microservices. This keeps all registrations, attendance and money records in one database, so related changes are saved together in a single transaction.

B. User Roles and Access Control

Ayojanix has four roles. Each role can use only its own features, as shown in Fig. 2.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

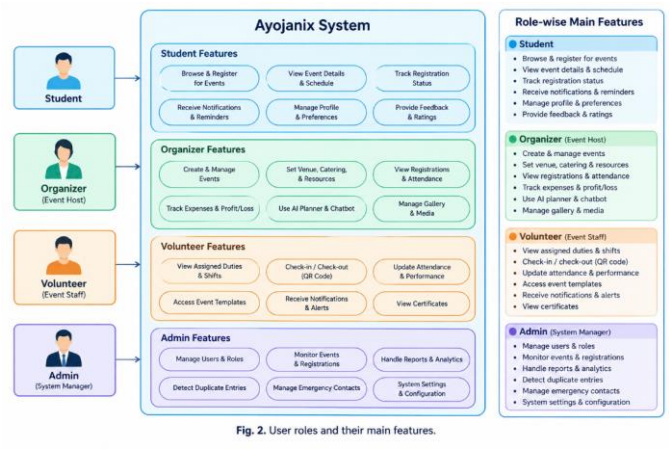


Fig. 2. User roles and their main features.

Admin: manages venues, vendors and user roles, and can see everything.

- **Organizer:** creates events, assigns volunteers, manages money records, uses the planning tools and issues certificates.
- **Volunteer:** scans QR passes at the gate.
- **Participant:** registers for events, gets a QR pass, gives feedback and downloads certificates.

Passwords are stored as bcrypt hashes with a cost factor of 10. After login, the server gives the user a JWT token signed with HMAC-SHA256, which expires after 7 days. The server checks the role on every protected request and returns HTTP 403 if the role is not allowed. The frontend also hides pages that a user cannot open, but the real protection is on the server.

C. Database Design

The database has 13 tables. Fig. 3 shows the main ones and how they are linked.

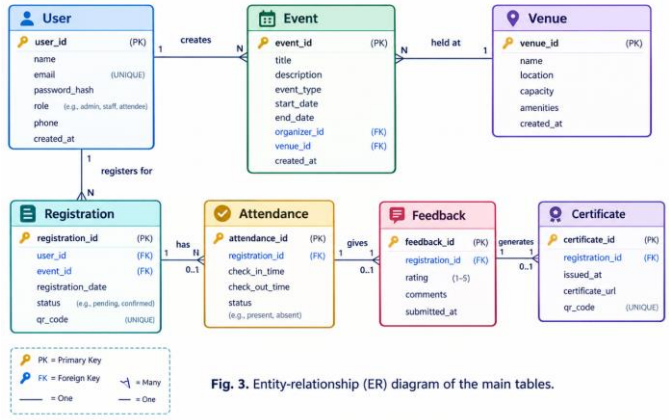


Fig. 3. Entity-relationship (ER) diagram of the main tables.

Important rules are enforced by the database itself:

- A student can register only once per event, using a unique key on (eventId, userId).
- Each registration code and certificate code is unique, which also makes searching them fast.
- Each registration can have only one certificate and one feedback entry.
- When an event is deleted, its registrations, attendance and feedback are deleted with it, so no orphan records remain.

Because these rules are in the database, they hold even if two requests arrive at the same moment.

IV. HOW THE MAIN FEATURES WORK

A. Event Registration

Each event moves through fixed stages: Draft → Published → Registration Open → Registration Closed → Ongoing → Completed. An event can also be Cancelled in the early stages.

A student's registration is accepted only if all of these are true:

- the event is open for registration,
- the registration deadline has not passed,
- the number of confirmed registrations is less than the event capacity, and
- the student has not already registered for this event.

The capacity check and the new registration are saved together in one database transaction. This means two students cannot both get the last seat, even if they click "Register" at the same moment.

After a successful registration, the server creates a unique code in the form **AYX-YYYY-NNNNN** and turns it into a QR code. The QR code uses **error correction level H**, so it can still be read if up to 30% of it is damaged, for example by a cracked phone screen. The QR image is stored as a Base64 string in the database, so no image files are saved on the server.

B. QR Check-in and Check-out

At the event, a volunteer opens the scanner page on a phone, selects the event and scans each student's pass. The server checks every scan in the steps shown in Fig. 4.

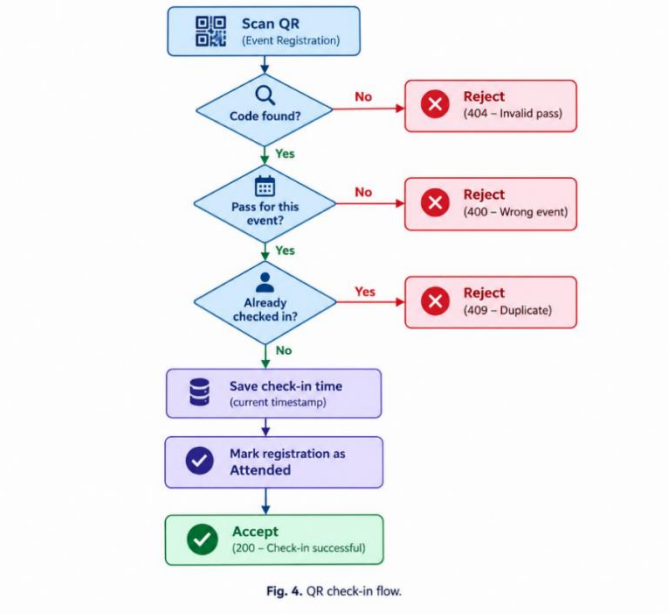


Fig. 4. QR check-in flow.

1. **Is the code valid?** If no registration has this code, the scan is rejected with HTTP 404.
2. **Is it the right event?** If the pass belongs to another event, the scan is rejected with HTTP 400.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

3. **Has it already been used?** If the student is already checked in and has not checked out, the scan is rejected with HTTP 409.
4. If all checks pass, the server saves the check-in time and marks the registration as **Attended**.

Step 3 is the **anti-passback rule**. If a student shares a screenshot of a pass, the second person is stopped, and the scanner shows a red alert with the time of the first entry. A successful scan shows a green screen with a beep, so the volunteer does not need to read the screen.

When the student leaves, a second scan at the exit saves the check-out time. The organizer can then find how long each student stayed:

Time spent = Check-out time − Check-in time

This helps organizers decide certificate eligibility for workshops that need full attendance. If a student's phone is not working, the volunteer can type the registration code manually, and the same checks are applied.

C. Food Demand Prediction

Colleges usually order one meal for every registered student, but not everyone attends, so extra food is wasted. Ayojanix estimates attendance first and orders meals based on that estimate.

Let **R** be the number of registered students and **η** the expected attendance rate in percent. By default η is 82%, and it is kept between 50% and 100%. For hackathons, η is set to at least 90%, because more registered students usually attend.

Predicted attendance: A = round(R × η / 100)

Meals to order (with 4% safety buffer): M = round(A × 1.04)

Veg meals = round(M × 0.70), Non-veg meals = round(M × 0.30)

The system also shows how many meals would be left over if the college ordered one meal per registration:

Expected surplus: W = R − A

Cost of surplus: W × ₹180 (default meal cost)

TABLE II. EXAMPLE FOR 500 REGISTRATIONS (H = 82%)

Item	Value
Registered students (R)	500
Predicted attendance (A)	410
Meals to order (M)	426
Veg / Non-veg meals	298 / 128
Surplus with one meal per registration (W)	90 meals (₹16,200)
Actual reduction in meals ordered (R − M)	74 meals (₹13,320)

The 82% default is only a starting value. As more events are run, it should be replaced with the real attendance rate recorded by QR check-ins.

D. Event Planning and Venue Matching

The planner helps organizers prepare an event. For **P** expected participants and a total budget **B**, it suggests:

- **Volunteers** = max(4, round(P / 25)), that is, 1 volunteer per 25 people with a minimum of 4.
- **Check-in desks** = max(1, round(P / 100)), that is, 1 desk per 100 people.
- **Venue:** from the available venues with capacity of at least P, it picks the one with the smallest capacity, so that a 40-person workshop does not book a 600-seat auditorium.
- **Budget split:** as shown in Table III.

TABLE III. DEFAULT BUDGET SPLIT

Category	Share
Catering and hospitality	35%
Venue and facilities	20%
Technical and audio-visual	15%
Prizes and mementos	15%
Kits, badges and printing	10%
Contingency	5%

For example, an event with 500 participants gets 20 volunteers and 5 check-in desks. These values are suggestions, and the organizer can change them.

E. Feedback Sentiment Analysis

After the event, students rate it and write a comment. The system finds the sentiment of each comment using a **lexicon-based method**:

1. The comment is changed to lowercase and split into words.
2. The words are matched against a list of positive words (such as "great", "helpful" and "informative") and a list of negative words (such as "boring", "late" and "noisy").
3. If there are more positive words, the comment is **Positive**, with score = min(0.99, 0.60 + 0.10 × positive count).
4. If there are more negative words, it is **Negative**, with score = max(−0.99, −0.60 − 0.10 × negative count).
5. If the counts are equal, it is **Neutral**, with a score of 0.10.

For example, "The workshop was great and very helpful" has 2 positive words and 0 negative words, so it is Positive with a score of 0.80. The results are shown on the organizer's dashboard.

F. Certificates and Verification

Certificates are issued only to students marked as **Attended**, and only one certificate per registration. Each certificate is created on the server as a PDF using the PDFKit library and sent directly to the browser. It has a unique certificate code and a QR code that links to a public verification page, as shown in Fig. 5.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026



Fig. 5. Certificate verification flow.

Anyone, such as an employer, can scan the QR code without logging in and see the student name, event name, issue date and status saved in the database. If someone edits the name on the PDF, it will not match the database record. An organizer can also mark a certificate as **Revoked**, for example in a case of misconduct, and the verification page will then show it as invalid.

G. Finance, Volunteers and Assistant

- **Finance:** every payment is saved as an income or expense entry linked to an event. When a student registers for a paid event, an income entry is created automatically. The event balance is total income minus total expenses. Vendor bills can be downloaded as PDF invoices.
- **Volunteers:** organizers assign each volunteer a duty, such as “Gate 1 Scanner”, and a time shift.
- **Assistant:** a role-aware query assistant answers common questions using live data, such as the number of active events or check-ins. If a Participant asks about money, it refuses, because finance data is only for Organizers and Admins.

V. IMPLEMENTATION

A. Tools and Technologies

Ayojanix is written in TypeScript for both the frontend and the backend, so both sides share the same data types. Table IV lists the main tools.

TABLE IV. TECHNOLOGIES USED

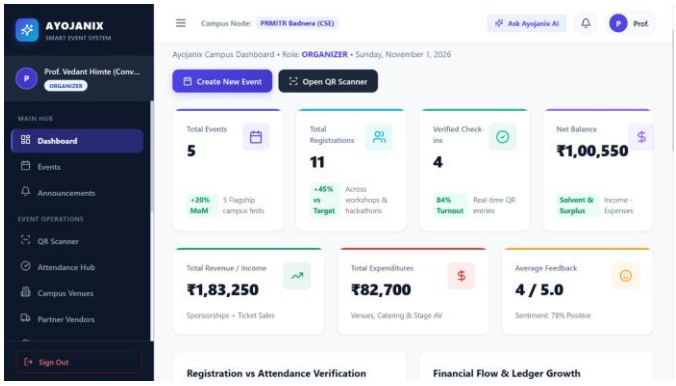
Part	Technology	Use
Frontend	React 18, Vite, TypeScript	Web interface and dashboards
Routing	React Router	Page navigation and role-based page guards
QR scanning	html5-qrcode	Reading QR codes through the phone camera
Charts	Recharts	Graphs on dashboards
Backend	Node.js, Express	REST API and business logic
Database	Prisma ORM with SQLite / PostgreSQL	Data storage and queries
Security	bcrypt, JWT, Helmet, express-rate-limit	Password hashing, login tokens, secure headers, request limits
QR generation	node-qrcode	Creating QR passes and certificate codes
PDF generation	PDFKit	Certificates and invoices

The project has two parts: a client folder for the React app and a server folder for the Express API, with one shared Prisma schema. During development, the frontend runs on port 5173 and the API on port 5000. For deployment, the frontend is built into static files served by Nginx, which also forwards API requests to the Node.js server managed by PM2.

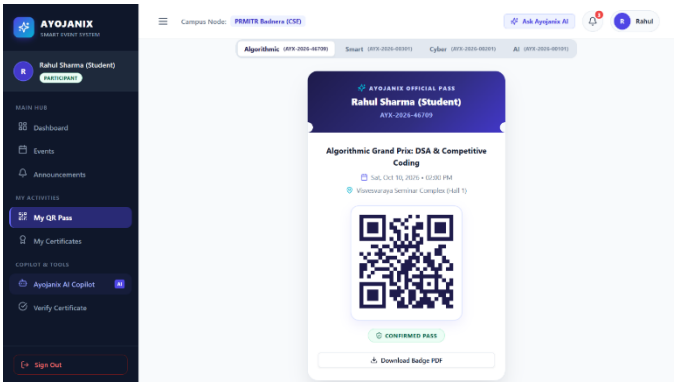
B. User Interface

The main screens are shown below.

Organizer dashboard. After login, each user sees a dashboard for their role. The organizer dashboard shows the number of events, registrations, check-ins and the money balance, with charts (Fig. 6).



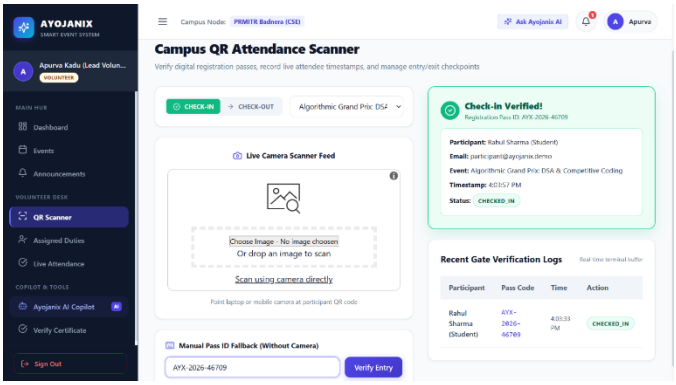
Student QR pass. After registering, the student sees their QR pass with the event name and registration code. The pass can be shown on the phone or printed (Fig. 7).



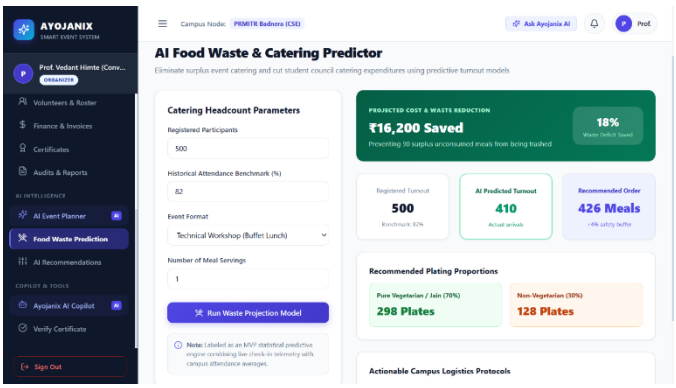
Volunteer scanner. The volunteer selects the event and points the phone camera at the pass. A valid pass shows a green screen with the student’s name (Fig. 8a). A pass that has already been used shows a red alert with the time of the first entry (Fig. 8b).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

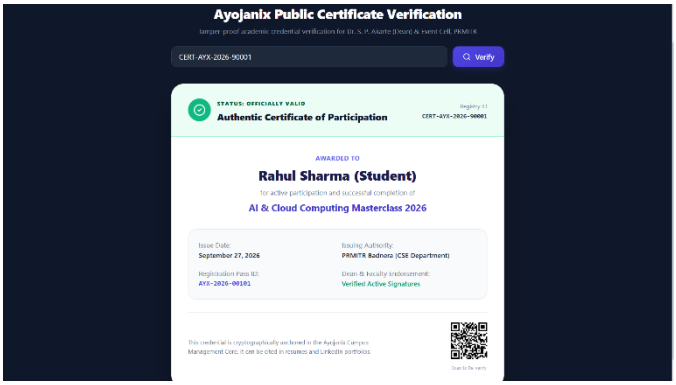
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026



Food prediction. The organizer enters the number of registrations and the expected attendance rate. The page shows the predicted attendance, meals to order, the veg/non-veg split and the expected surplus (Fig. 9).



Certificate and verification. The student downloads a PDF certificate with a QR code. Scanning the QR code opens the public verification page, which shows whether the certificate is valid (Fig. 10).



VI. Testing and Results

A. Test Setup

The system was tested in a local development setup. The backend ran on Node.js with an SQLite database, and the frontend ran in a web browser. The database was filled with demo data using the seed script: sample users for all four roles, sample events, registrations and a certificate. The QR scanner was tested using both the camera scan option and the manual pass ID entry.

B. Functional Testing

We tested the main features of the system by running the cases in Table V and comparing the actual result with the expected result.

TABLE V. FUNCTIONAL TEST RESULTS

No.	Test case	Expected result	Actual result	Status
1	Check-in with a valid pass for the selected event	Pass accepted and status set to CHECKED_IN	Pass AYX-2026-46709 accepted, status shown as CHECKED_IN (Fig. 8a)	Pass
2	Scan the same pass again	Rejected as a duplicate	“Verification Denied”, with the message that the participant was already checked in and the time of first entry (Fig. 8b)	Pass
3	Check-in by typing the pass ID manually	Same checks as a camera scan	Pass ID entered in the manual field was verified (Fig. 8a)	Pass

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

4	Food prediction for 500 registrations at 82% attendance	410 predicted, 426 meals, 298 veg and 128 non-veg	Same values shown on the prediction page (Fig. 9)	Pass
5	Verify a valid certificate code	Certificate shown as valid with name, event and date	CERT-AYX-2026-90001 shown as valid with all details (Fig. 10)	Pass

All five test cases gave the expected result.

C. Discussion

The tests show that the main rules of the system work as designed. The most important result is test 2: when the same pass was scanned a second time, the system refused entry and showed the time of the first check-in. This is the rule that stops one QR pass from being shared among several students. Test 3 shows that a volunteer can still check in a student whose phone is not working, using the same checks. Test 4 confirms that the food prediction page gives the same values as the formulas in Section IV-C. Test 5 shows that anyone can confirm a certificate is genuine by entering or scanning its code, without logging in.

D. Planned Evaluation

The current tests check that the system works correctly. They do not yet measure its performance on live events. As future work, we plan to:

- measure the average check-in time per student with QR scanning and compare it with a paper attendance list,
- compare predicted attendance with actual QR check-ins on real college events to find the error of the food prediction model, and
- measure usability with the System Usability Scale (SUS) [7], using feedback from organizers, volunteers and students.

VII. LIMITATIONS AND FUTURE WORK

A. Limitations

Ayojanix works well for its main purpose, but it has some limits:

1. **Needs internet.** The scanner checks every pass on the server, so it does not work offline. A poor network at a large venue can slow down check-in.
2. **Simple rules, not learning.** The sentiment analysis only counts words from fixed lists, so it misreads phrases like “not good” and cannot handle comments in Hindi, Marathi or Hinglish. The planning ratios are also fixed and do not learn from past events.
3. **Depends on default values.** Food prediction uses a default 82% attendance rate and a 70:30 veg/non-veg split until enough real event data is collected.
4. **Certificates have no digital signature.** Verification depends on the college database, so it fails if the server is down.
5. **Basic finance only.** The system records income and expenses but has no online payment gateway, so fees are entered by hand.
6. **Limited testing.** The system has so far been tested only with demo data in a development setup. It has not yet been evaluated on live events.

B. Future Work

1. **Offline scanning:** save the event’s pass list on the volunteer’s phone before the event and sync check-ins when the internet returns.
2. **Better prediction:** use attendance data from past events to set the attendance rate for each event type automatically.
3. **Better sentiment analysis:** handle negation words and support Indian languages, or train a small model once enough labelled feedback is collected.
4. **Signed certificates:** add a digital signature so that certificates can be verified even without the server.
5. **Online payments:** connect a UPI or payment gateway to record fees automatically.
6. **Multi-college use:** allow several colleges in a university to use one system while keeping their data separate.

VIII. CONCLUSION

This paper presented the design, implementation and testing of Ayojanix, a web-based event management system for colleges. It replaces separate tools such as Google Forms, paper attendance sheets and spreadsheets with one platform built on React, Node.js, Express and Prisma ORM. QR-based check-in with server-side checks stops invalid, wrong-event and duplicate passes, which prevents proxy entry. A rule-based decision support module helps organizers estimate food demand, plan volunteers, budget and venue, and understand feedback. Certificates can be verified by anyone through a QR code.

REFERENCES (IEEE STYLE)

[1] D. A. Shah, H. Vasudavan, and N. F. Razali, “Event management systems (EMS),” *Journal of Applied Technology and Innovation*, vol. 7, no. 2, pp. 45–53, 2023.

[2] R. N. Kulkarni, H. Sahana, B. S. Vastrad, E. Akshitha, and R. Jeevitha, “Novel approach to manage events in the organizations,” *International Research Journal of Engineering and Technology*, vol. 9, no. 6, pp. 1132–1136, Jun. 2022.

[3] G. Thombare, P. Jadhav, S. Dhere, P. Jadhav, and K. N. Honwadkar, “College event management: A survey of analytics and personalization,” *International Research Journal of Modernization in Engineering, Technology and Science*, vol. 5, no. 11, pp. 343–347, Nov. 2023, doi: 10.56726/IRJMETS45827.

[4] M. J. Q. Mayan-ao *et al.*, “Development of events management system,” *International Journal of Engineering Research & Technology*, vol. 14, no. 12, Dec. 2025.

[5] M. N. A. M. Sakeri *et al.*, “Digitization trends: The growing demand for smart event management systems among small entrepreneurs in developing regions,” *Journal of Information and Knowledge Management*, vol. 15, Special Issue, 2025, doi: 10.24191/jikm.v15iSI1.6097.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)