Online Bonafide Certificate Management System: A Multi-Level Approval Workflow with Automated PDF Generation and Email Notification

Tamilselvi K.[1], Monisha V.[2], Priya R.[3], Sivaranjani K.[4]

[1]Assistant Professor, Adhiyamaan College of Engineering, Hosur

[2,3,4]UG Scholars, Adhiyamaan College of Engineering, Hosur– 635109, India

***Abstract*— A student who needs a bonafide certificate for a scholarship, a bank loan, a passport or an internship usually has to carry a paper request from one desk to the next: the tutor, the Head of Department (HOD), the Principal and finally the college office. Each step depends on the person being present, the request cannot be tracked, and a lost or delayed form means the student starts again. The Online Bonafide Certificate Management System is proposed to close this gap. A student logs in with a registration number and date of birth, fills a short request form stating the purpose, and submits it. The request then moves through a fixed chain of role-based approvals (tutor, HOD, Principal), and each approver sees only the requests that belong to their stage. After the final approval, the office staff generate the certificate as a PDF with the college header, the student's details and the date, and the student is notified by email and can download the certificate from the dashboard. The system is built as a React (v18) front end that calls a Node.js and Express back end through REST requests, with a MySQL database that stores student details, requests, staff records, status and history. The status of every request is visible to the student at all times, and a certificate can be generated only after every required approval has been recorded. The prototype was exercised through all five role portals and produced the certificate shown in this paper. Quantitative measurements of turnaround time and user satisfaction have not yet been collected and are reported as limitations.**

Index Terms*— *bonafide certificate, student services, multi-level approval, workflow automation, React, Node.js, Express, MySQL, PDF generation, e-governance

I. INTRODUCTION

A bonafide certificate is a short document in which an institution confirms that a person is a genuine student of a particular course and academic year. It is requested often and for ordinary reasons: scholarship forms, education loans, bus and railway concessions, passport applications, visas, internships and competitive examinations. The document itself is only a few lines long. The process of obtaining it is not. In most colleges the student writes or collects a request form, takes it to the tutor, then to the HOD, then to the Principal, and finally to the office where the certificate is typed, stamped and signed. Every step requires the right person to be present, and the student has no way of knowing where the request is stuck.

Three problems follow from this. First, the process is slow and unpredictable, because a single absent signatory stops the whole chain. Second, it is opaque: the student cannot tell whether a request is waiting, approved or rejected without walking to the next desk. Third, it creates paperwork for the college itself, since each request produces a form, a register entry and a typed certificate that must be filed and later searched.

The Online Bonafide Certificate Management System is built on a simple observation: the request, the approvals and the certificate should belong to one workflow that everyone can see. In the proposed system a student submits one online request. The request is passed in a fixed order to the tutor, the HOD and the Principal, each of whom can accept or reject it with a single action. When the last approval is in place, the office generates the certificate as a PDF, and the student receives an email and can download the file from the dashboard. Every request carries a status that changes as it moves, so that the student, the approvers and the office all read the same record.

The work was carried out as an undergraduate mini project, and the claims are limited accordingly. The software workflow, from the student request to the generated certificate, was built and demonstrated through all five role portals. Measured turnaround times, load tests and user surveys have not yet been collected. The contributions of this paper are as follows.

1. A single online workflow in which the student request, the tutor, HOD and Principal approvals and the office certificate generation share one backend, one database schema and one status history.

2. A role-based access design with five user roles, in which the student signs in with registration number and date of birth and the staff roles sign in with email and password, so that each person sees only the tasks that belong to their role.
3. A rule that a certificate can be generated only after every required approval has been recorded, which prevents a certificate from being issued on an incomplete chain.
4. Automated PDF certificate generation with email notification, so that the student receives the final document without visiting the office.
5. A description of the prototype, with screen-level results and a plain statement of what has and has not yet been measured.

The remainder of the paper is organised as follows. Section II reviews related work and compares the proposed system with the manual process. Section III describes the system architecture and workflow. Section IV presents the module design. Section V reports results from the prototype. Section VI discusses them together with security and privacy considerations, Section VII lists limitations and the proposed evaluation protocol, and Section VIII concludes.

II. RELATED WORK AND COMPARISON

A. Web-based certificate generation

Jhanani et al. [1] describe a web-based system built on ASP.NET that lets students request a bonafide certificate online and also supports education-loan management. Their work shows that moving the request from paper to a web form is practical for a college and that the certificate itself can be produced from stored student data. The system is built on a Microsoft stack and ties the certificate to the loan process, whereas the present work focuses only on the bonafide request and uses an open-source JavaScript stack.

B. Student clearance and approval workflows

Eweoya et al. [2] present a web-based solution for student clearance in higher education, a process with the same structure as a bonafide request: a student must obtain sign-off from several offices in turn. Their work supports the view that an online, role-based sequence of approvals reduces the walking and waiting that paper clearance involves. The present system applies the same idea to a shorter chain in which each approver sees only the requests that have reached their stage.

C. User-centred design of e-certificate systems

Huliman et al. [3] design a website-based student e-certificate system using the User-Centered Design (UCD) method. Their emphasis is on usability: the interface is shaped by what students and administrators need to do. This is relevant to a bonafide system, because the people who use it most, students and busy staff, will adopt it only if each task takes a few clicks. The present system follows the same aim by giving each role a separate, minimal portal.

D. Gap and the proposed contribution

The reviewed works show that online certificate generation, online approval chains and usable interfaces are each feasible. The descriptions reviewed say little about three points that matter in a college setting: a fixed approval order that ends with a separate office stage for generating the certificate, a status that the student can follow at every step, and automatic delivery of the final PDF by email. The proposed system combines these in one workflow. Table I compares the manual process with the proposed system. The entries describe the process itself and do not report measured values.

TABLE I. COMPARISON OF THE MANUAL PROCESS AND THE PROPOSED SYSTEM

Aspect	**Manual (paper) process**	**Proposed online system**
Request	Paper form carried by the student	Online form submitted from any device
Approval order	Student walks between desks	Fixed order: tutor, HOD, Principal, office
Status visibility	None until the student asks	Status shown on the student dashboard
Skipping a stage	Possible	Not possible; each stage unlocks the next
Certificate	Typed and stamped by hand	Generated as a PDF from stored data
Delivery	Collected at the office	Email notification and dashboard download
Records	Paper register	Searchable database with status history

III. SYSTEM ARCHITECTURE AND WORKFLOW

A. Architecture overview

Fig. 1 presents the architecture of the system, which has four layers. The users, shown on the left of the figure, are the student, the staff (tutor), the HOD, the Principal and the office. The front end is a React (v18) web application that uses React Router DOM for navigation, Axios for API calls, HTML2PDF.js for PDF generation and SweetAlert2 for alerts and notifications. It sends API requests to the back end, which is built on Node.js with Express.js. The back end handles API requests, performs authentication and authorisation, validates and processes the data, applies the business logic, generates the PDF and sends the email notification. It stores and retrieves records in a MySQL database that holds the student details, the certificate requests, the staff, HOD, Principal and office records, and the status and history of every request. External services are used for email delivery (Nodemailer or Resend) and for PDF generation (HTML2PDF.js).

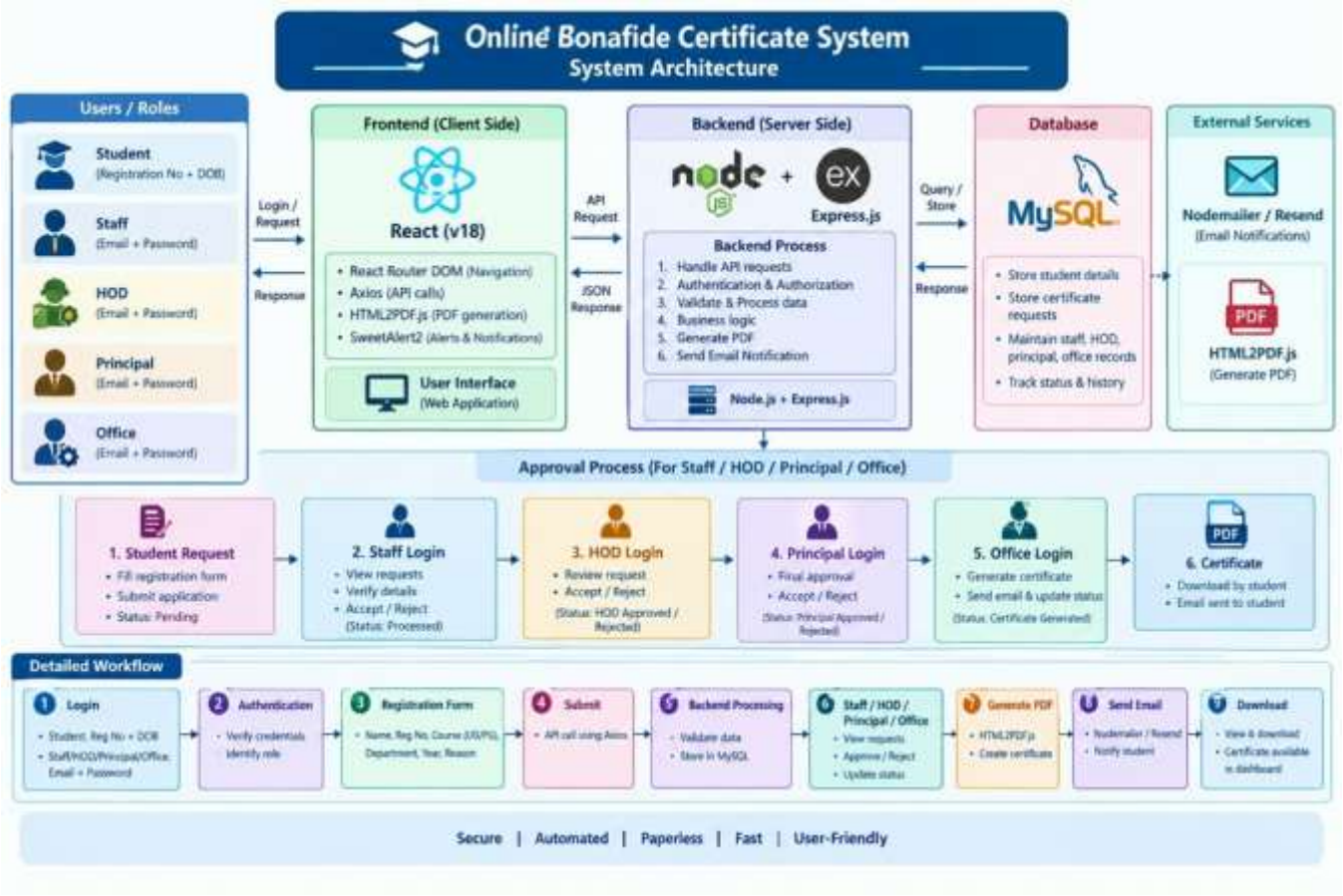


Fig. 1. Online Bonafide Certificate System architecture and workflow.

The front end and the back end are separated deliberately. The browser holds only the interface and talks to the server through a small set of REST endpoints [8], so the database is never reachable from the client and the same back end could serve a mobile application later.

B. User roles and authentication

The system has five roles. A student logs in with the registration number and date of birth, which are already known to the college, so no separate student password has to be created or remembered. The staff (tutor), HOD, Principal and office users log in with an email address and a password. After login the back end identifies the role and returns only the portal that belongs to it. Fig. 2 shows the portal home page, from which each role enters its own login: Student Login to request a bonafide certificate, Tutor Login to approve student requests, HOD Login for departmental approval, Principal Login for Principal approval, and Office Login to generate and print the certificate.

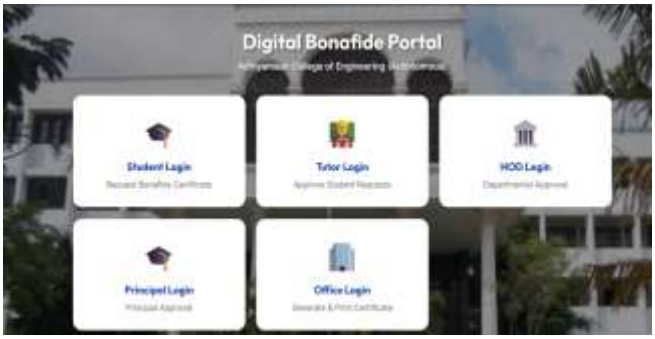


Fig. 2. Digital Bonafide Portal showing the five role logins.

C. Approval process

The approval process is a fixed chain, shown in the lower part of Fig. 1 and summarised in Table II. The student fills the registration form and submits the application, and the request is stored with the status Pending. The tutor logs in, views the requests, verifies the details and accepts or rejects each one; an accepted request moves on with the status Processed. The HOD reviews the request and accepts or rejects it, giving the status HOD Approved or Rejected. The Principal gives the final approval, giving the status Principal Approved or Rejected. Only then does the request reach the office, where the certificate is generated, the email is sent and the status is updated to Certificate Generated. The student then downloads the PDF, and a copy is also sent by email.

TABLE II. STAGES OF A SINGLE REQUEST

Step	Actor	Action	Resulting status
1	Student	Fills registration form and submits	Pending
2	Tutor	Views request, verifies details, accepts or rejects	Processed / Rejected
3	HOD	Reviews request, accepts or rejects	HOD Approved / Rejected
4	Principal	Gives final approval, accepts or rejects	Principal Approved / Rejected
5	Office	Generates certificate, sends email, updates status	Certificate Generated
6	Student	Downloads certificate from dashboard; receives email	—

The order is deliberate. A request cannot reach the HOD before the tutor has accepted it, and it cannot reach the office before the Principal has approved it. A rejection at any stage ends the request, and the student can see the stage at which it stopped. This gives the student the one piece of information the manual process never provides: where the request is and why it is not moving.

D. Detailed workflow

The detailed workflow at the bottom of Fig. 1 has nine steps. (1) Login: the student uses the registration number and date of birth, and the staff roles use email and password. (2) Authentication: the back end verifies the credentials and identifies the role. (3) Registration form: the student enters name, registration number, course (UG/PG), department, year and the reason for the request. (4) Submit: the front end sends the request to the API using Axios. (5) Backend processing: the data are validated and stored in MySQL. (6) Staff, HOD, Principal and office users view the requests, approve or reject them and update the status. (7) Generate PDF: the certificate is created with HTML2PDF.js. (8) Send email: Nodemailer or Resend notifies the student. (9) Download: the student views and downloads the certificate, which is also available in the dashboard.

E. Request processing procedure

Algorithm 1 summarises how the back end handles a request at any approval stage. The order of the checks is deliberate: the cheap checks on the request come first, the database lookups follow, and the email is sent last because a failed email must never undo an approval that has already been recorded.

ALGORITHM 1. HANDLING AN APPROVAL ACTION

1. Receive action A (accept or reject) for request R from an authenticated user.
2. If the user's token is missing or invalid, return error 401.
3. If the user's role is not the role allowed to act on the current status of R, return error 403.
4. If R does not exist, return error 404.
5. If A is reject, set the status of R to Rejected at the current stage and store the reason.
6. If A is accept, set the status of R to the next status in the chain (Processed, HOD Approved, Principal Approved).
7. Record the user, the stage and the timestamp in the history table.
8. If the new status is Principal Approved, make R visible to the office role.
9. Return the updated request (HTTP 200); send the notification email when the certificate is generated.

IV. MODULE DESIGN AND IMPLEMENTATION

The development followed five phases: requirement analysis, system design, module implementation, testing and validation, and deployment and evaluation. Requirement analysis identified the users (students,

tutors, HODs, Principals and office staff) and the need for online applications, multi-level approval, status tracking and PDF generation. System design produced the architecture, the database tables, the user interfaces and the approval workflow. The system was then implemented as the following modules.

A. Student registration and login module. Authenticates students using the registration number and date of birth. The credentials are checked against the student records in MySQL, and a failed login returns an error without revealing which field was wrong.

B. Bonafide request module. Allows the student to submit an application by stating the purpose of the certificate. The form collects name, registration number, course, department, year and reason, and the back end stores the request with the status Pending.

C. Multi-level approval module. Manages sequential approval by the tutor, the HOD and the Principal. Each approver's dashboard lists only the requests whose current status matches their stage, which keeps the lists short and prevents out-of-order action.

D. Application tracking module. Displays the current approval status of each request to the student, so the student can see whether it is Pending, Processed, HOD Approved, Principal Approved, Rejected or Certificate Generated.

E. Certificate generation module. Generates the downloadable PDF after all required approvals are complete. The certificate carries the college header with logo and accreditation details, the date, the heading "Bonafide Certificate", a statement confirming that the named student with the given registration number, son or daughter of the named parent, is studying in the stated year and course during the stated academic year, the AICTE approval reference, and the Principal signature block, as shown in Fig. 3.

F. Office management module. Allows authorised office staff to see the approved requests, generate the certificate, print it and send it by email.

G. Notification module. Supports email notifications that update the student about the progress and the final certificate. Email is sent through Nodemailer or Resend from the back end.

Testing and validation. Each module was tested to confirm that requests are submitted correctly, that approvals follow the defined sequence and that certificates are generated only after all required approvals. Usability, security and reliability were then checked on the running application in a development environment.

V. RESULTS

A. Application screens

Fig. 2 shows the home page of the Digital Bonafide Portal of the college. The five role cards make the process easy to follow: the student enters through Student Login to request a certificate, the tutor through Tutor Login to approve student requests, the HOD through HOD Login for departmental approval, the Principal through Principal Login for Principal approval, and the office through Office Login to generate and print the certificate. The same layout is used for all roles, with only the destination changing.

B. Generated certificate

Fig. 3 shows a certificate generated by the system. It follows the format of the college's manual certificate: the header with the institution's name, status and affiliation, the date, the heading "Bonafide Certificate", the certification text with the student's name, registration number, parent's name, year, programme and academic year, the AICTE approval reference, and the signature block for the Principal. The screen offers two actions, Print / Save as PDF and Send to Mail, which correspond to the PDF generation and email notification steps of the workflow. The student details in the sample certificate come from the authors' own test data.



Fig. 3. Bonafide certificate generated by the system.

C. Workflow verification

The workflow was exercised from the student request to the generated certificate through each of the five portals. Table III lists the behaviours that were checked on the prototype.

TABLE III. FUNCTIONAL CHECKS ON THE PROTOTYPE

Check	Expected behaviour	Result
Student login	Registration number and date of birth accepted; wrong values rejected	Verified
Request submission	Request stored with status Pending	Verified
Tutor stage	Only Pending requests listed; accept moves request to Processed	Verified
HOD stage	Only tutor-approved requests listed	Verified
Principal stage	Only HOD-approved requests listed	Verified
Out-of-order action	Request cannot skip a stage	Verified
Certificate generation	PDF produced only after Principal approval	Verified
Email notification	Student notified when certificate is generated	Verified
Status tracking	Student sees current status on dashboard	Verified

D. Comparison with the manual process

In the manual process the student must be present at four desks, and the request is a single paper that can be lost. In the proposed system the student acts once, at submission, and then follows the request from the dashboard. The number of approvals is unchanged, because the college's authorisation rules remain the same; what changes is that approvers can act from their own screens without the paper being handed over. A numerical comparison of turnaround time would require the measurements described in Section VII, and no figure is claimed here.

VI. DISCUSSION

A. Observations

Three points stand out from the prototype. First, a fixed chain with a status is what removes most of the friction. The approvals themselves are the same as on paper, but because every stage is a database state, the next approver sees the request as soon as the previous one acts, and the student sees the same state. Second, role-based portals keep each screen small. A tutor never sees Principal actions, and an office user never sees requests that are still waiting for approval, so the risk of acting on the wrong request is low. Third, separating the approval from the certificate generation was useful. Approvers decide; the office only produces the document. This keeps the decision with the authorised staff and the clerical task with the office.

B. Security and privacy considerations

A bonafide request contains the student's name, registration number, date of birth, course and the stated purpose, which together form a personal record. The system limits access by role, so each user sees only the data needed for their task. Staff passwords should be stored using salted hashing rather than plain text [7], and a production deployment needs encrypted transport (HTTPS), token-based authentication with expiry and an audit trail of approvals. Because the student login uses the registration number and date of birth, which are not secret, a production system should add a second factor, such as a one-time password sent to the registered email address, before a student can view or download a certificate. Generated certificates should carry a unique serial number or a QR code that the receiving organisation can use to verify the document, so that a copy cannot be altered unnoticed.

VII. LIMITATIONS AND NEXT MEASUREMENTS

A. Known limitations

The limits of this work are stated so that the results are not read as more than they are.

1. The system has been demonstrated on a prototype with test data. No turnaround time, number of concurrent users or response time has been measured.
2. Student login uses the registration number and date of birth only. This is convenient but weak as authentication, and a second factor is needed for real use.
3. Security is at prototype level. Encrypted transport, token expiry, rate limiting on login and an approval audit log must be in place before real student data are stored.
4. The certificate does not yet carry a verification code, so an authenticity check by the receiving organisation is not possible.
5. Email delivery depends on an external service. If the service is unavailable, the student can still download the certificate from the dashboard, but the notification is delayed.
6. The system has not been evaluated by students and staff, so usability claims are limited to the design intention.

B. Proposed evaluation protocol

Three experiments are planned. The first measures the end-to-end turnaround time, from the moment a student submits a request to the moment the certificate is available, for a sample of at least thirty requests, and compares it with the same measurement for the manual process in the same department. The second is a load test in which a fixed number of simulated students submit requests at once, reporting the median and 95th percentile response time of the API and the number of requests lost or rejected. The third is a usability survey of students and staff using a standard instrument such as the System Usability Scale (SUS), reported with the number of respondents. Only after these three have been run can the system be described as validated for anything beyond a demonstration.

VIII. CONCLUSION AND FUTURE WORK

The Online Bonafide Certificate Management System joins the student request, the tutor, HOD and Principal approvals, the office certificate generation and the email notification into a single workflow. By automating the process and using role-based access, it reduces paperwork, lets the student follow the status of a request at every step and ensures that a certificate is issued only after all required approvals. The prototype is built on React, Node.js with Express and MySQL, with HTML2PDF.js for the certificate and Nodemailer or Resend for email, and it was demonstrated through all five role portals. The measurements of turnaround time, load and usability are the next thing to prove.

The next steps are to carry out the three experiments in Section VII, to add a one-time-password second factor and a QR-code verification on each certificate, to add SMS or WhatsApp alerts, to support other certificate types (conduct, course completion, transfer) on the same workflow, and to move the system from a development environment to a secured, hosted deployment.

ACKNOWLEDGMENT

The authors thank Mrs. K. Tamilselvi, Assistant Professor, Department of Computer Science and Engineering, for her guidance as supervisor, the Head of the Department, the Principal, and the faculty of Adhiyamaan College of Engineering for the facilities and support provided during this work.

REFERENCES

[1] M. Jhanani, M. Prakasam, M. Poornima, and B. Abirami, “Student Online Bonafide Certificate Generation and Education Loan Management System Using ASP.NET,” *Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE)*, vol. 9, no. 5, May 2020, doi: 10.17148/IJARCCE.2020.9530.

[2] I. O. Eweoya, A. O. Agbeyangi, and J. M. Lukose, “Streamlining Student Clearance in Higher Education: A Web-based Solution,” *Information Impact: Journal of Information and Knowledge Management*, vol. 16, no. 1, pp. 47–62, 2025, doi: 10.4314/iijikm.v16i1.5.

[3] Huliman, R. Wijaya, and Ayu, “Design of a Website-Based Student E-Certificate System Using the User Centered Design (UCD) Method,” *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 5, no. 1, pp. 1064–1069, 2025, doi: 10.59934/jaiea.v5i1.1551.

[4] Meta Platforms, “React documentation.” [Online]. Available: https://react.dev

[5] OpenJS Foundation, “Node.js documentation” and “Express.js documentation.” [Online]. Available: https://nodejs.org and https://expressjs.com

[6] Oracle Corporation, “MySQL reference manual.” [Online]. Available: https://dev.mysql.com/doc

[7] OWASP Foundation, “Password storage cheat sheet,” *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org

[8] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000.