International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Judicial Case Management System (JCMS): A Transparent, Human-in-the-Loop Approach to Priority Scoring, Timeline Prediction and Fairness-Aware Hearing Scheduling

Karan B Gaddemalle[1], Preetham S Gowda[2], Kanasu V[3], Bhuvana P Bharadwaj[4]

[1]Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur, Karnataka, India
[2]Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur, Karnataka, India
[3]Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur, Karnataka, India
[4]Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur, Karnataka, India

Corresponding author: karanbgaddemalle@gmail.com

Abstract – Court backlogs delay justice, and the delay weighs most on people held in custody before trial. This paper presents JCMS, a Judicial Case Management System that helps a court registry decide who should be heard first, estimates when a case will end, and tells citizens about their hearings. JCMS combines a transparent fixed-weight priority score (P-Score) that always places undertrial prisoners first, two estimators of case duration with SHAP explanations and calibrated 80% ranges, and a weekly hearing scheduler whose output stays a draft until a registrar approves it, with a written reason for every override. A fairness-quota variant protects a share of slots for the longest-waiting cases. Citizens receive e-mail alerts confirmed by a six-digit one-time code and recorded in a tamper-evident log. All evaluation uses synthetic data and is reported as such. In a 52-week simulation, P-Score scheduling cuts the mean wait of undertrial cases from 168 to 4 days but leaves 48 of 750 old cases unheard; a 10% quota removes that starvation. The duration models beat a median baseline by only 2.7 to 4.0%, because most variation is noise. We discuss limits and the path to real data.

Keywords: **judicial case management, case prioritization, court scheduling, explainable prediction, fairness, undertrial prisoners.**

I. INTRODUCTION

Indian courts carry a very large case backlog. Replies given in Parliament, drawing on the National Judicial Data Grid [1], put the number of cases pending in district and subordinate courts at about 4.99 crore (49.9 million) in July 2026 [2]. Delay is not only an efficiency problem. Prison Statistics India 2022 counted 434,302 undertrial prisoners among 573,220 inmates, that is 75.1% of all prisoners [3]: people who have not been convicted of anything and wait in custody for their case to be heard. The Law Commission of India has argued for years that arrears must be treated as a systemic problem of capacity and process, not as a series of individual delays [4].

Registries must decide every week which cases are listed before which judge. Two kinds of tools could support that decision without replacing the judge. An *ordering* tool explains why one case should come before another. A *prediction* tool gives a litigant a realistic range for the end of the case. Both carry risk. A score that nobody can inspect can hide bias [5], and a schedule that only chases urgency can leave cases that are never urgent without a hearing for years.

This paper describes **JCMS**, a Judicial Case Management System built by a four-member student team and tested end to end on synthetic data. Four design principles run through it. (i) *The system recommends and a human decides*: every schedule is a draft until a registrar or judge approves it, and every override needs a written reason. (ii) *Transparency*: the priority formula has fixed, published weights and every score is stored with its breakdown. (iii) *Honesty about uncertainty*: estimates are shown as ranges and every number in this paper is labelled with the data it came from. (iv) *Privacy*: the public portal returns only whitelisted fields.

The contributions are: (1) a transparent priority score with a forced undertrial rule; (2) two duration estimators with SHAP explanations and calibrated 80% ranges, evaluated with a time-based split and an explicit noise ceiling; (3) a greedy hearing scheduler and a fairness-quota variant, compared by simulation with first come, first served (FCFS), including the cases where the priority score does worse; (4) a citizen portal with e-mail alerts confirmed by a one-time code and stored in an append-only, hash-chained log; and (5) a full-stack implementation with automated tests, an accessibility audit and a reproducible evaluation. Section II reviews related work, Section III describes the method, Section IV gives the results, Section V discusses them and Section VI concludes.

II. LITERATURE REVIEW

A. Backlog and Delay in Courts

The Law Commission of India separates pendency, delay, arrears and backlog, and finds no single objective formula for the "normal" time a case should take [4]. It also links delay to the number of judges and to vacancies, which is why our synthetic generator lets judge workload and the court's vacancy rate shift case durations. Official dashboards such as NJDG [1] publish counts of pending cases; they do not rank cases or estimate durations, which is the gap JCMS addresses.

B. Machine Learning on Judicial Data

Much published work applies machine learning to the *outcome* of cases. Aletras et al. predicted decisions of the European Court of Human Rights from the text of judgments [6], and Katz et al. built a general model for the behaviour of the United States Supreme Court [7]. In the sources we reviewed, predicting *how long* a case will take received less attention than predicting how it will be decided, and we did not find an open, end-to-end system that combines duration estimates with scheduling and citizen notification. We therefore treat our results as a methodological demonstration on synthetic data, not as a comparison with published accuracy figures.

C. Explanations and Calibrated Uncertainty

SHAP assigns each input a contribution to one prediction [8], and a fast exact version exists for tree ensembles [9]. We use it so that a registrar and a citizen can see, in days, why an estimate is long or short. Gradient boosting with LightGBM [10] and ridge regression from scikit-learn [11] provide the models. Ranges come from quantile regression [12] followed by a conformal calibration step in the spirit of conformalized quantile regression [13], so that the stated 80% range actually covers about 80% of unseen cases.

D. Fairness, Accountability and Human Oversight

Reporting on the COMPAS risk scores found unequal error rates between groups in an algorithm used in criminal justice, a problem that only careful evaluation reveals [5]. Fairness criteria such as treating similar individuals similarly [14] and work comparing machine predictions with human bail decisions, which shows that the two can differ systematically and must be evaluated carefully [15], shaped our rule that the schedule is a draft, with reasons recorded for overrides. For the alert log we use the tamper-evident hash-chain idea from secure audit logging [16].

E. Priority Scheduling and Starvation

Priority scheduling in operating systems is known to starve low-priority jobs, and *aging*, which raises the priority of jobs that have waited, is the classic remedy [17]. The P-Score contains an age term that behaves like aging, but we show in Section IV that it limits starvation only partly. We therefore add an explicit protected share of slots.

F. Positioning

In short, official dashboards report counts, research systems predict outcomes, and scheduling theory warns about starvation. JCMS joins these in one auditable pipeline of score, estimate, schedule and notify. It does not claim a better predictor; its contribution is the combination, the explicit safeguards, and an honest evaluation of where each part helps and where it does not.

III. METHODOLOGY

A. System Architecture

JCMS has four layers (Fig. 1). A React front end serves two route trees: a public portal in five languages (English, Hindi, Kannada, Tamil and Telugu) and an admin app for registrars, judges and administrators. A Node.js and Express API, with Prisma as the database layer, implements authentication with JSON Web Tokens and three roles, case and hearing management, the P-Score, scheduling, subscriptions and notifications. A Python FastAPI service loads the trained models and returns predictions with explanations. PostgreSQL stores cases, hearings, predictions, schedule drafts, subscriptions and the alert log. If the ML service is down, the API marks predictions as stale and never invents a number. The whole system runs on one laptop with Docker; a hosted deployment (static hosting for the front end and a managed Node platform for the API) is planned but has not been done. Besides the ranked queue and the case page with its scores, the admin app offers a printable full case record (parties, advocates, case history) in the layout familiar from the eCourts case status view, kept separate from our recommendations.

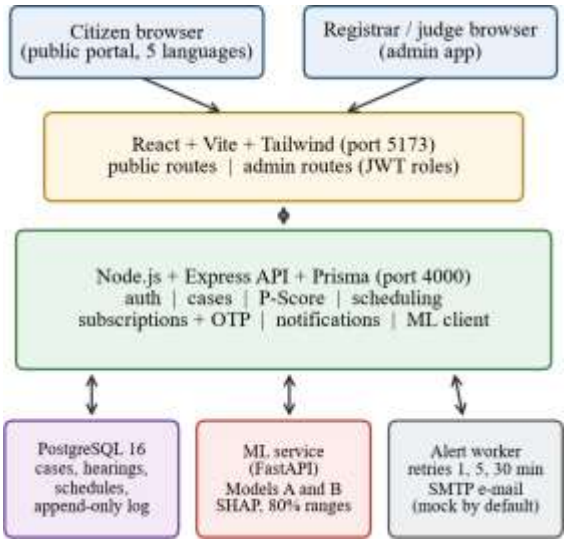


Fig. 1. JCMS architecture: four layers and the alert worker.

B. Data

No real court records were available to us, so all models are trained and tested on synthetic cases from a seeded generator whose duration is a formula plus log-normal noise ($\sigma = 0.6$), with a long right tail, 8% quick disposals and effects for case type, court tier, judge workload and vacancy (Table I). The generator lives in the repository and every case carries an *isSynthetic* flag. Splits are by filing year, so the models are tested on later cases than they were trained on. Cases filed in the test years that are still pending are missing from the test set, which we state as a censoring limitation. A second, small sample of 2,000 cases in seven courts around Tiptur, Karnataka, drives the live demonstration; the model training data are unchanged by it.

TABLE I. SYNTHETIC DATA AND SPLITS

Item	Value
Training universe	100,000 cases, 15 courts, 6 states
Disposed cases used	73,104 (219,312 snapshots)
Train (filed up to 2013)	33,541 cases
Validation (2014–2015)	8,762 cases
Test (2016–2018)	12,880 cases
Still pending, test years	15.0%
Demo sample (Tiptur region)	2,000 cases, 7 courts, 32,441 hearings

C. The Priority Score (P-Score)

Each pending case receives a score from 0 to 100 built from four normalised parts with fixed weights (Table II and (1)): age A in days d, category C, adjournments J for a adjournments, and vulnerability V.

$$P = 100 \times (0.30A + 0.25C + 0.20J + 0.25V) \quad (1)$$

$$A = min(d/3650,1),\ J = 1 - min(a/10,1) \quad (2)$$

TABLE II. P-SCORE COMPONENTS

Part	Weight	Normalised value
Age A	0.30	min(d/3650, 1)
Category C	0.25	undertrial 1.0; criminal 0.8; family 0.6; civil, writ 0.5; other 0.3
Adjournments J	0.20	1 − min(a/10, 1)
Vulnerability V	0.25	min(1, 0.5 custody + 0.2 senior + 0.2 disabled + 0.3 minor victim)

Bands are CRITICAL (80 or more), HIGH (60 to 79.99), MEDIUM (40 to 59.99) and LOW. One rule overrides the formula: any case with an undertrial in custody is placed in the CRITICAL band and sorted ahead of every other case, and undertrials are ordered among themselves by P-Score. Ties are broken by case number (CNR), so the same input always gives the same order. The weights are policy choices recorded in one configuration file; the tests lock the defaults, and changing them needs the court's agreement, not the developers'. Every score is stored with its component breakdown so that a registrar can see what produced it. As a worked example (Fig. 2), the top case of the demonstration queue is an undertrial in custody, filed 1,613 days ago, with three adjournments and two vulnerability flags (accused in custody 0.5, minor victim 0.3, so V = 0.8): 100 × (0.30 × 0.442 + 0.25 × 1.0 + 0.20 × 0.7 + 0.25 × 0.8) = 72.3, made of 13.3 + 25.0 + 14.0 + 20.0 points.



Fig. 2. Registrar case page: P-Score breakdown of the top-ranked case in the queue.

D. Duration Estimators

Model A predicts total duration and number of hearings at filing, and days and hearings remaining for a case in progress, using ridge regression and LightGBM; a delay-risk classifier predicts whether a case will be slower than the median of its type. The model for each task was chosen on the validation years only. Inputs known when the case is filed are case type, court tier, state, filing year, judge workload and the court's vacancy rate; in-progress models add case age, hearings held, adjournment share and hearings per year. **Model B** answers the question "how many days remain?" from the P-Score, case type, court tier, state, workload, age and adjournments, with the P-Score computed only from information known at the snapshot date (no leakage).

Training protocol. Each disposed case yields three mid-case snapshots at different ages, with every feature computed only from information known at the snapshot date, and all snapshots of a case stay on the same side of the split. Categorical inputs (case type, court tier, state) are one-hot encoded, and an unseen category becomes all zeros, so a new state cannot break a prediction. Models are fitted on the training years, compared on the validation years and scored once on the test years, which never influence a choice. Durations are right-skewed, so models work on a log scale. We compare the median-by-case-type baseline, ridge regression, random forest, gradient boosting and LightGBM.

Ranges. Two LightGBM quantile models give the 10th and 90th percentile of the outcome. On the validation years we then measure how far outcomes fall outside their ranges and widen or narrow every range on the log scale so that 80% of validation cases fit. *Explanations.* SHAP values give the effect of each input in days; the portal shows the five largest in plain words. *Baseline.* The reference is the median duration for the case type; "gain" is the reduction in mean absolute error (MAE) against it. We also compute a *noise ceiling*: the error of the generator's own formula, which no model trained on filing-day inputs can beat by much. With n test cases, true values y_i, predictions p_i and ranges $[l_i, u_i]$, we report the mean absolute error

(4), the gain over the baseline (5) and the coverage of the ranges (6).

$$MAE = (1/n)\sum|p_i - y_i| \quad (4)$$

$$gain = 1 - MAE/MAE_{base} \quad (5)$$

$$coverage = (1/n)\sum[l_i \le y_i \le u_i] \quad (6)$$

E. Hearing Scheduler and Human Oversight

For one court and one week, the greedy scheduler takes each judge's pending cases in priority order (undertrials first, then P-Score, ties by CNR) and fills the judge's slots day by day. A worked example: a judge has 2 slots on Monday and Tuesday and cases A (undertrial, 50), B (90), C (70), D (60) and E (30); the order A, B, C, D, E gives Monday A, B, Tuesday C, D, and E waits. Sorting costs O(n log n) for n cases. Because every case takes exactly one slot, taking the best remaining case first maximises the total P-Score of the chosen cases, so more elaborate optimisation would matter only with unequal case lengths or extra constraints.

The **fairness-quota scheduler** reserves a share ρ (default 10%, allowed 0 to 50%) of each judge's free slots. The remaining slots go to the top-ranked cases as before; the protected slots go to the cases that waited longest since their last hearing among those the ranking would not reach; everything chosen is then placed in priority order so urgent cases still get the earliest days. With ρ = 0 it equals the greedy scheduler, which the tests check. The P-Score formula is not changed.

Judicial discretion. A generated schedule is always saved as a DRAFT. Moving or removing an entry requires a written reason of at least five characters, and the system records who, why and when. Approval is an explicit human action, it writes the hearing dates in one all-or-nothing database transaction, and approving twice changes nothing. There is no automatic approval (Fig. 3).

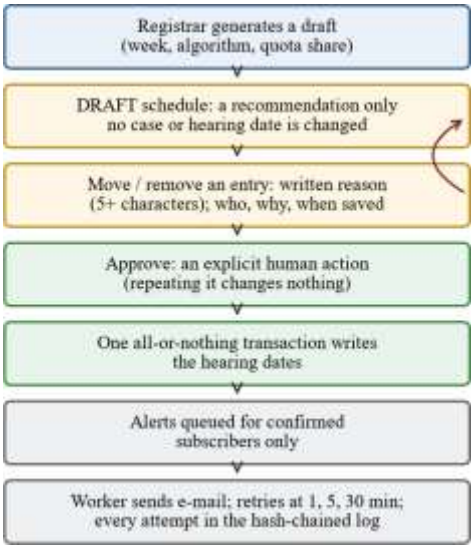


Fig. 3. Schedule lifecycle, from draft to confirmed-subscriber alert.

F. Citizen Alerts, One-Time Codes and the Audit Log

Citizens sign up for e-mail alerts on a case page. *Confirmation.* A new sign-up is stored switched off and a random six-digit code is e-mailed; the person types it into the site, and only then do alerts start. Following the spirit of the guidance on one-time codes [18] (short life, limited tries, rate limits and single use), the code is valid for 10 minutes, allows five wrong tries, can be re-sent once a minute, and each IP address is limited to 20 attempts an hour. Only a keyed hash (HMAC-SHA-256) of the code and the sign-up identifier is stored, and a used code is erased. Wrong, expired and unknown codes give the same answer, and repeating a sign-up for a confirmed address changes nothing, so the API reveals nothing about who is subscribed.

Delivery and log. A worker sends queued alerts and retries failures after 1, 5 and 30 minutes; each message has an idempotency key so that the same event is never queued twice. Every attempt is written as a new row to an append-only log that a database trigger protects against updates and deletes. Rows form a hash chain (3), where H is SHA-256 and r_i holds the id, case, event type, status, message body and time of row i, so changing an old row breaks every later hash and the integrity check in the admin app finds it.

$$h_i = H(h_{i-1} \| r_i) \quad (3)$$

Contact details are masked in every response and log. The public API returns only whitelisted fields: vulnerability flags, custody status and P-Scores never leave the server, and party names in family cases or cases with a minor victim are replaced by "Party A" and "Party B". E-mail is the only delivered channel; SMS code exists but is not offered, and real messages are sent only when the system is deliberately switched to live mode.

G. Security and Access Control

Staff sign in with a password that is stored only as a salted hash and receive a JSON Web Token that expires after eight hours. Three roles apply: an administrator sees all courts, while a registrar or judge can read and change only cases of their own court (other courts return an authorisation error), administrators and registrars can change hearings, and schedules can be generated and approved by registrars, judges and administrators, again within their own court. All request bodies and parameters are validated against schemas before use, and every response uses one envelope for data and errors. Public routes are rate limited (60 lookups a minute, five sign-ups an hour and 20 code attempts an hour per IP address, with separate limits on sign-in), cross-origin access is limited to the configured website address, and responses that concern sign-ups are not cached. Secrets are read from environment variables that are never committed, and the server refuses to start in production with a placeholder or short signing secret.

H. Scheduling Simulation

To compare policies we simulate one court for 52 weeks with 5 judges and 150 hearings a week. The starting backlog is 750 pending cases (150 per judge) resampled from synthetic cases, and about 8.5 new cases arrive per week (Poisson), a load of 1.1 times capacity. A case returns after at least four weeks, a hearing is adjourned with probability 31.4%, and a case cannot be disposed before its third hearing, then with probability 5.7% per hearing; these values come from the synthetic hearing table. All policies see the same backlog, new cases and hearing outcomes for a given seed, and they differ only in the order passed to the same scheduler. We run 20 seeds (42 to 61) and report mean and standard deviation. *Wait* is the time from filing to the first hearing for cases filed in the first 26 weeks; a case never heard counts to the end of the year, so waits are lower bounds. FCFS means oldest filing first, ignoring custody. The fairness quota is evaluated on the same worlds.

I. Verification

Code is verified at four levels: unit and HTTP tests (Vitest for JavaScript, pytest for Python), a scripted end-to-end run against the real database and ML service, a full-stack smoke test that also checks privacy and rate limits, and an automated accessibility audit (axe-core, WCAG 2.1 levels A and AA [19]) of 13 public and admin page states, including the code-entry step. Reports are produced by commands that anyone can re-run, and a result is written down only if the code that computed it ran.

IV. RESULTS

Every number in this section was measured on synthetic data. They show how the methods behave inside the synthetic world; they are not evidence about real courts.

A. Duration Estimators

Table III shows the test-year results. At filing, ridge regression is chosen for total duration (MAE 636.8 days against 663.1 for the baseline, a gain of 4.0%, R^2 = 0.040); for cases in progress LightGBM is chosen (456.9 against 472.4, 3.3%). The gains are small, and the noise ceiling explains why: on the validation years the generator's own formula reaches an MAE of 727.2 days against the baseline's 755.4, so only 3.7% of the baseline error is learnable from filing-day inputs. The small gains are therefore a property of the data, not a failure of the models. The delay-risk classifier reaches an accuracy of 0.584 at filing (majority class 0.532, ROC-AUC 0.626) and 0.750 for cases in progress (ROC-AUC 0.821).

TABLE III. DURATION ESTIMATORS, TEST YEARS, MAE IN DAYS OR HEARINGS

Task	Model	MAE	Base	Gain
Total days (filing)	Ridge	636.8	663.1	4.0%
Hearings (filing)	Ridge	8.5	8.7	2.2%
Days left (in progress)	LightGBM	456.9	472.4	3.3%
Hearings left	LightGBM	6.0	6.2	2.5%
Days left, Model B	LightGBM	459.6	472.4	2.7%

The 80% ranges are honest but wide. After calibration the test coverage is 82.1% at filing (median width 2,477 days) and 81.6% in progress (1,615 days), against a target of 80%; before calibration it was 80.8% and 80.9%. Coverage is lowest for the rare OTHER case type (72.5% at filing, 356 test cases). Misses are lopsided (12.6% of cases are faster than the range and 5.3% slower), which is the signature of censoring: the slowest test-year cases are still pending and so are missing. The models lean towards short estimates: 45% of cases are predicted too long, and the mean signed error is −153 days at filing. For a citizen this means the range, not the single date, is the honest message.

Table IV breaks the at-filing results down by slice. The gain over the baseline is positive in every case type and court tier, and largest in subordinate courts (11.7%), but coverage is below the 80% target for family cases (78.1%), for the rare OTHER type and for subordinate courts (79.1%): conformal calibration targets 80% overall, not per group. Across judge-workload quartiles MAE rises from 580 to 705 days while coverage stays between 80.4% and 83.9%.

TABLE IV. ERROR BY SLICE AT FILING, TEST YEARS

Slice	MAE	Base	Gain	Cover
Civil	717	747	4.0%	83.9%
Criminal	627	654	4.1%	82.5%
Family	511	537	4.8%	78.1%
Other	447	460	2.9%	72.5%
Writ	649	657	1.2%	82.1%
District court	651	670	2.8%	82.8%
High court	651	659	1.2%	81.8%
Subordinate	559	634	11.7%	79.1%

SHAP shows what drives the predictions (Table V): case type, judge workload and judge vacancies dominate at filing, and case age joins them for cases in progress. These are descriptions of the model, not causes in real courts.

TABLE V. MEAN ABSOLUTE SHAP IMPACT (DAYS)

Factor	At filing	In progress
Case type	195	81
Judge workload	141	58
Judge vacancies in court	126	46
Case age	–	47
Court tier	83	6
Hearings held so far	–	16
State	21	6

Does the P-Score help to predict duration? In an ablation, removing the P-Score from Model B raises the test MAE by only 0.61 days (95% interval 0.15 to 1.13 from 500 resamples of whole cases), and replacing it by the raw vulnerability flags changes it by 0.23 days

(interval −0.21 to 0.59, including zero). The P-Score is built from age, category and adjournments, which the model already receives, so it mostly repeats them. We conclude that the P-Score earns its place as a *priority tool*, not as a duration predictor. Table VI lists the Model B bake-off: LightGBM was chosen on the validation MAE (520.4), and no learner improves on the baseline by more than 2.7%.

TABLE VI. MODEL B BAKE-OFF, MAE IN DAYS

Model	Validation	Test
Median by case type (baseline)	529.5	472.4
Ridge	530.4	469.3
Random forest	521.4	460.8
Gradient boosting	522.2	459.6
LightGBM (chosen)	520.4	459.6

B. Scheduling Policies

Table VII and Fig. 4 compare the policies. P-Score scheduling is a large improvement for urgent cases: undertrial new cases wait 4.1 days for a first hearing against 168.3 under FCFS, and the share of new cases not heard within 26 weeks falls from 36.1% to 7.3%. Every band of new cases waits less on average. The price is paid by old, low-priority cases. Under P-Score, 48.0 of the 750 backlog cases (6.4%; worse than FCFS in all 20 seeds) receive no hearing in the whole year, against none under FCFS, and the longest wait of a LOW-band new case rises from 250.8 to 338.0 days. Fewer cases are finished (402.8 against 422.4), because the P-Score spends slots on new cases and a case cannot finish before its third hearing in this model.

TABLE VII. POLICIES IN THE SIMULATION (MEAN OF 20 SEEDS)

Measure	FCFS	P-Score	+10% quota
Undertrial wait, days	168.3	4.1	4.1
LOW mean wait, days	162.5	83.0	78.0
LOW longest wait, days	250.8	338.0	141.3
New cases unheard at 26 wk	36.1%	7.3%	0.0%
Old cases never heard (of 750)	0	48.0	0.0
Cases disposed in the year	422.4	402.8	404.4

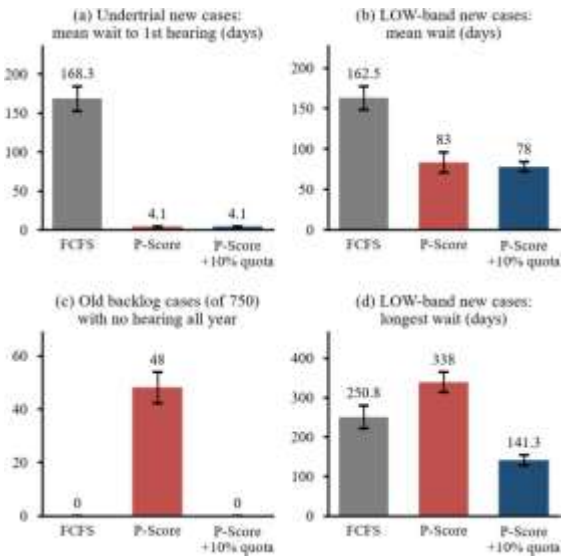


Fig. 4. FCFS, P-Score and P-Score with a 10% protected quota (mean and standard deviation of 20 seeds).

Role of the age term. With the age weight set to zero, 136.4 ± 13.7 old cases are never heard (against 48), so the age term does protect old cases, yet new low cases then wait only 7 days on average. The reason is the adjournment part: a case with no adjournments receives full points, which every new case has, so a new civil case with no flags scores 32.5 while a typical old backlog case (1,190 days, five adjournments) scores 32.3. Only the 9.8 age points keep the old case in the race. The age term does not prevent starvation; it decides who is starved.

C. Fairness Quota

A protected 10% share removes the starvation in the main scenario: no old case goes unheard, the longest LOW wait falls from 338.0 to 141.3 days (FCFS: 250.8) and no new case waits more than 26 weeks, while undertrials still wait 4.1 days. A 5% share already reaches zero unheard old cases (longest wait 177.1 days), and 20% shortens the longest wait to 93.0 days but gives more of the week to low-ranked cases, so the share is a policy choice for the court. Disposals stay about the same (404.4 against 402.8) and do not recover the FCFS level (422.4). A sensitivity analysis over backlogs of 100, 150 and 200 cases per judge and loads of 1.0 to 1.2 shows that the backlog matters more than the load. When the court keeps up (100 per judge) all policies give short waits and nobody is starved; at 150 the quota removes the starvation at every load; in a heavily overloaded court (200 per judge) the quota cannot create capacity: 24.0% of new cases still wait more than 26 weeks, against 41.1% without it.

TABLE VIII. QUOTA SHARE SWEEP, MAIN SCENARIO (MEAN OF 20 SEEDS)

Share	Unheard	LOW max	LOW mean	>26 wk	Disposed
0% (P-Score)	48.0	338.0	83.0	7.3%	402.8
5%	0.0	177.1	88.6	1.2%	404.9
10%	0.0	141.3	78.0	0.0%	404.4
20%	0.0	93.0	60.6	0.0%	405.1

TABLE IX. OTHER COURTS: P-SCORE ALONE → WITH 10% QUOTA

Scenario	Unheard	LOW max	>26 wk
100 per judge	0.1→0.0	69.8→35.9	0.2→0.0%
150, load 1.0	39.3→0.0	329.4→134.5	5.1→0.0%
150, load 1.2	55.5→0.0	341.4→146.2	7.9→0.0%
200 per judge	159.6→0.0	367.0→239.4	41.1→24.0%

Tables VIII and IX give the details: the share of old cases never heard (Unheard, of 750), the longest and mean wait of LOW-band new cases in days (LOW max, LOW mean), the share of new cases not heard within 26 weeks (>26 wk) and the cases disposed. Cases disposed are almost unchanged by the share, and the shortest worst-case waits come with the largest share.

D. Implementation and Verification

The full system runs on a laptop with Docker. Table X summarises the checks. The batch job that refreshes predictions for all 450 pending cases of the demonstration data completes in about two seconds with no failures. On the Tiptur-region sample the registrar sees 193 pending cases of the Tiptur court (23 of them undertrials), the 23 undertrials are listed first, and a non-undertrial case scoring 68.4 comes after undertrials scoring as low as 52.1, as the rule intends (Fig. 6, Fig. 2). Across the 450 pending cases the P-Score ranges from 24.2 to 72.3 (mean 41.6) with 54 CRITICAL, 3 HIGH, 169 MEDIUM and 224 LOW cases. Almost nothing reaches HIGH (Fig. 5) because, without the undertrial rule, scores top out near 72; we report this rather than tune the thresholds.

A working week at the Tiptur court. Generating a draft for the week of 2 November 2026 produces 180 scheduled hearings over the court's three judges, 21 of them CRITICAL, with none left unscheduled; the greedy and the quota algorithms give the same summary counts here because every pending case fits, so the protected share has nothing to reassign, which is why the simulation, not this small court, is where the policies differ. In the end-to-end run the registrar removes an entry with a written reason, the summary counts update to match the list, approval writes the hearing dates and queues alerts, and the worker's alert is recorded as SENT in the hash-chained log, whose integrity check then passes. Real e-mails, sent in separate runs to team-owned inboxes only, were logged the same way. A sign-up whose code was never entered received nothing.

Lessons from testing. Clicking through the real stack in a browser found a defect that the mocked tests had hidden: after a registrar moved or removed a schedule entry, the schedule screen went blank, because the API returned only the changed entry while the page and its mock server expected the whole schedule. A second check found that the summary tiles were computed once at generation and no longer matched the list after a removal. Both were fixed, and the end-to-end script now asserts the corrected behaviour, so the fix is locked by a test.

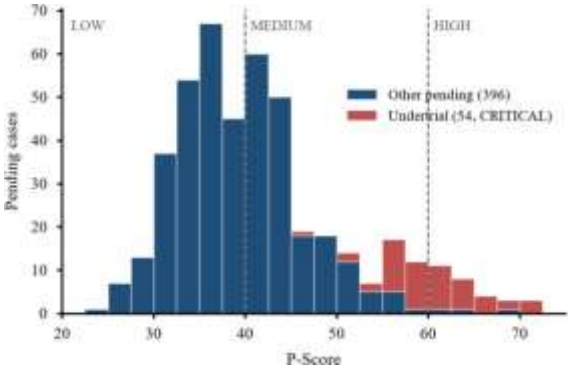


Fig. 5. P-Score of the 450 pending demo cases; the 54 undertrials in custody are forced into the CRITICAL band whatever their score.

TABLE X. AUTOMATED CHECKS

Check	Result
Back-end tests (Vitest)	427 passed, 5 skipped
ML tests (pytest)	157 passed
Front-end tests (Vitest)	107 passed
Simulation tests	37 passed
End-to-end script (real DB, ML)	18 of 18 steps
Full-stack smoke test	26 of 26 checks
Accessibility (axe, WCAG 2.1 A/AA)	0 violations, 13 page states
Predictions for pending cases	450 updated, 0 failed



Fig. 6. Citizen portal: estimate with range and plain-language reasons.

V. DISCUSSION

Table XI traces each design principle to the mechanism that implements it and to the evidence in this paper.

TABLE XI. DESIGN PRINCIPLES, MECHANISMS AND EVIDENCE

Principle	Mechanism	Evidence
Human decides	DRAFT status; written reason; explicit approval	End-to-end steps; tests
Transparency	Fixed weights; stored breakdown; plain reasons	Table II; Fig. 6, Fig. 2
Honest uncertainty	80% ranges; noise ceiling; stale flag	Tables III, IV; 82.1%, 81.6%
Privacy	Field whitelist; masking; hashed codes; masked logs	Smoke test: no private fields
Auditability	Append-only hash-chained log	Integrity check passes

What the results show. A transparent score can put urgent cases first at almost no cost in complexity, and the simulation quantifies the benefit (4 against 168 days for undertrials). The same simulation shows that urgency-first ordering is not neutral: in an overloaded court it moves the waiting from new urgent cases to old non-urgent ones. A fairness quota is a simple, explainable remedy, but it is a trade-off that belongs to the court: a larger protected share helps the longest-waiting cases and takes slots from high-ranked ones. FCFS is not fair either, because in an overloaded court new cases, including undertrials, are the ones that starve (12.4 undertrials per seed had no hearing within 26 weeks).

What the models do not show. The duration estimators improve on a median baseline by 2.7 to 4.0%, and the noise ceiling says why. We do not claim accuracy for real courts, and we make no comparison with published accuracy figures, because those use real data, different courts and different definitions of delay. The explanations describe the model; they do not identify causes of delay. The model leans towards optimistic estimates, which is why the portal shows a range and a plain-language disclaimer.

Fairness and open questions. The adjournment term penalises a litigant for adjournments that the court itself may have caused; the data model already holds the number of party-caused adjournments for a fairer variant, but the default follows the project specification and the team has not settled the question. Karnataka shows the largest systematic tilt among states in the slice analysis (mean signed error −193 days at filing), but each state rests on one to four synthetic courts, so this reflects those courts, not the state. The weights of the P-Score are policy, and we cannot show that they are right.

Limitations. (1) All data are synthetic; durations come from a formula plus noise, so models recover the generator's rules. (2) The simulation covers one court and one year, a fixed four-week return gap that makes CRITICAL hearings arrive in waves, no judge absences and no human step that could add a starved case. (3) The Hindi, Kannada, Tamil and Telugu texts of the portal were drafted by AI and have not been reviewed by native speakers, and alert e-mails are in English. (4) The automated accessibility audit passed on all audited pages and a scripted keyboard check of the admin screens passed, but no hands-on screen-reader test was carried out. (5) Alerts were tested with real e-mail to team-owned addresses only; behaviour at scale is untested. (6) The test set misses the slowest cases (censoring).

Threats to validity. Internally, the simulation assumes that the chance of disposal does not depend on case type, priority or age and that a case returns after a fixed gap; the CRITICAL waves in the weekly load are partly an artefact of that gap. Externally, one synthetic court and one year say nothing certain about real courts. On the construct side, the wait to the first hearing is only one fairness measure; it does not capture harm to a litigant or the value of a hearing. Statistically, 20 seeds give stable means (standard deviations are reported), but the ranking of policies may change with other assumptions, which is why the sensitivity analysis varies backlog and load.

Ethical considerations. Placing undertrials first is a value judgement in favour of liberty, not a technical result, and the protected quota is a second one; both belong to the judiciary and should be reviewed by it. Estimates can anchor expectations, so the portal states that they are estimates and not legal advice. No real personal data were used, public pages mask sensitive parties and hide vulnerability data, and the system keeps a human responsible for every decision.

Path to real use. Before any real use, the system needs research access to court data through the institution or a legal-research partner, a generator or model recalibrated on real aggregates, the same fairness evaluation repeated per court, an independent audit, and a pilot with registry staff. Until then it remains a decision-support prototype whose method and safeguards are real and tested and whose numbers describe synthetic data.

VI. CONCLUSION

JCMS shows that prioritisation, duration estimation, hearing scheduling and citizen notification can be combined in one system that keeps a human in charge and states its uncertainty. On synthetic data, a fixed-weight P-Score reduces the wait of undertrial cases from 168 to 4 days, and the plain greedy scheduler starves 48 of 750 old cases; a protected 10% quota removes that starvation without changing the formula. The duration models are only slightly better than a baseline, which the noise ceiling explains. Future work is to validate the method on real data with court partners, to test an age term based on time since the last hearing, to let courts choose the protected share per week, and to complete the language review and screen-reader testing.

DATA AND CODE AVAILABILITY

The data are synthetic and are generated by a seeded generator that is part of the project source code, which is available from the authors on request. The commands that regenerate the data, train the models, run the scheduling simulation (20 seeds) and the fairness-quota evaluation, and the test suites are documented with the code, and each reported number can be reproduced with them. No real court records or personal data were used.

ACKNOWLEDGEMENT

The authors thank their guide, Mrs. Shilpa, Assistant Professor, Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur, for guidance and support, and Dr. Maithri C, Professor and Head of the Department, and Dr. Sateesh

Kumar H C, Principal, for providing the facilities for this work. All cases, names and numbers used in this study are synthetic (invented); no real court record or person is represented.

REFERENCES

[1] National Judicial Data Grid, eCourts Services, Department of Justice, Government of India. [Online]. Available: https://njdg.ecourts.gov.in/ (accessed Oct. 27, 2026).

[2] "Jammu Kashmir High Court pendency rises to 44,551 cases; subordinate courts cross 3.65 lakh, Centre tells Lok Sabha," Kashmir Life, Jul. 2026. [Online]. Available: https://kashmirlife.net/ (accessed Oct. 27, 2026).

[3] National Crime Records Bureau, Prison Statistics India 2022. New Delhi, India: Ministry of Home Affairs, Government of India, 2023.

[4] Law Commission of India, "Arrears and backlog: Creating additional judicial (wo)manpower," Rep. No. 245, Jul. 2014.

[5] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "Machine bias," ProPublica, May 23, 2016.

[6] N. Aletras, D. Tsarapatsanis, D. Preoţiuc-Pietro, and V. Lampos, "Predicting judicial decisions of the European Court of Human Rights: A natural language processing perspective," PeerJ Comput. Sci., vol. 2, p. e93, 2016.

[7] D. M. Katz, M. J. Bommarito II, and J. Blackman, "A general approach for predicting the behavior of the Supreme Court of the United States," PLoS ONE, vol. 12, no. 4, p. e0174698, 2017.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 30, 2017, pp. 4765–4774.

[9] S. M. Lundberg et al., "From local explanations to global understanding with explainable AI for trees," Nat. Mach. Intell., vol. 2, no. 1, pp. 56–67, Jan. 2020.

[10] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 30, 2017, pp. 3146–3154.

[11] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[12] R. Koenker and G. Bassett, "Regression quantiles," Econometrica, vol. 46, no. 1, pp. 33–50, 1978.

[13] Y. Romano, E. Patterson, and E. J. Candès, "Conformalized quantile regression," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 32, 2019, pp. 3543–3553.

[14] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in Proc. 3rd Innov. Theor. Comput. Sci. Conf. (ITCS), 2012, pp. 214–226.

[15] J. Kleinberg, H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan, "Human decisions and machine predictions," Q. J. Econ., vol. 133, no. 1, pp. 237–293, 2018.

[16] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," ACM Trans. Inf. Syst. Secur., vol. 2, no. 2, pp. 159–176, May 1999.

[17] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, 10th ed. Hoboken, NJ, USA: Wiley, 2018.

[18] P. A. Grassi et al., "Digital identity guidelines: Authentication and lifecycle management," NIST Special Publication 800-63B, Jun. 2017 (updated Mar. 2020).

[19] A. Kirkpatrick, J. O Connor, A. Campbell, and M. Cooper, Eds., "Web Content Accessibility Guidelines (WCAG) 2.1," W3C Recommendation, Jun. 2018.