UGANDA CHRISTIAN UNIVERSITY
A Centre of Excellence in the Heart of Africa

FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY
DEPARTMENT OF COMPUTING AND TECHNOLOGY
EASTER 2025 SEMESTER **EXAMINATION**

PROGRAM: *DIPLOMA IN INFORMATION TECHNOLOGY*

YEAR: 2 SEMESTER: 1

COURSE CODE: *CSC1202*

COURSE NAME: *WEB & MOBILE APPLICATION DEVELOPMENT*

EXAMINATION TYPE: *100% PROJECT-BASED EXAM*

PROJECT DURATION: DECEMBER 2025

TIME ALLOWED: *TWO WEEKS*

Examination Instructions

1. **Uganda Christian University Examination Guidelines and Policies:**
   - The general Uganda Christian University examination guidelines and academic & financial policies apply to this examination. Violating any of the policies by the student automatically makes this examination attempt void, even if you have completed and submitted the answer booklet.
2. **Project Duration and Submission:**
   - This exam consists of a project to be executed in **Two weeks.**
   - Assessment of the project shall be based on **five milestones**, evaluated during the duration of the project. Each milestone shall be evaluated out of **20 marks**.
   - At the end of the project, the following **SHALL** be submitted on **Moodle**:
     - A well-written project report (Font: Trebuchet MS, 12Pts, 1.5 spacing, justified alignment), using **IEEE Referencing style**.
     - Two links to **GitHub repositories** (Frontend and Backend). These repositories should have evidence of **collaboration** and **consistent individual commits.**
3. **Group Work and Individual Contribution:**
   - Every student is responsible for proving their contribution towards every milestone. Marks may be awarded to every student individually based on their contribution.
   - Each group must consist of **6 students.** Collaboration and communication within the group are essential for the successful completion of the project.

PART A: PROJECT THEME

Build a secure, auditable E-Voting System that supports:

1. Nominations (candidate onboarding & approvals)
2. Voter verification (eligibility validation + one-time access)
3. Voting (secret ballot, double-vote prevention)
4. Reports (turnout, per-position results, audit log)

Target platform: Web (React) or Mobile (React Native) with a backend of your choice (Node/Express, Django). DB: PostgreSQL/MySQL.

Roles & users to support

- **Election Admin** (sets positions, windows, eligibility rules, views full audit)
- **Returning Officer** (approves nominations, oversees tally)
- **Candidate** (submits nomination, sees approval status)
- **Voter** (verifies eligibility, casts exactly one secret ballot)

Functional requirements (exam scope)

1. **Nominations**

   a. Create positions (e.g., Guild President, Secretary) with nomination windows.
   b. Candidate self-registration: name, program, manifesto (PDF), photo; validation.
   c. Returning Officer approval/rejection with reason; notifications.
   d. Candidate list per position, status tracking, immutable **nomination audit trail**.

2. **Voter verification**

   a. Import or register an **eligibility list** (CSV).
   b. Verify voter via unique ID + **OTP** (email/SMS or in-app pin); rate-limiting.
   c. On successful verification: issue **single-use ballot access** (token/session). Prevent re-verification once a ballot is cast.
3. **Voting**

   a. **Secret ballot**: ballot does not store voter identity; no PII in vote records.
   b. One-vote-per-position; prevent double voting & replay (token invalidation). Election window enforcement (not before start / after close).
   c. Basic method: **First-Past-The-Post** per position.

4. **Reports & dashboards**

   a. Live **turnout** (eligible vs verified vs voted).
   b. Per-position tallies; spoiled/invalid ballots count.
   c. Export **CSV/PDF** (turnout, results by position, nominations list, full audit log).

d. Admin dashboard with filters (position, time range).

Non-functional requirements

1. **Security:** hashed secrets, HTTPS assumption, JWT/opaque session tokens, CSRF protection (web), input validation, OWASP basics, minimal PII in logs, least-privilege roles.
2. **Auditability:** append-only **AuditLog** records for key actions (nomination decisions, verification, ballot issued, ballot cast, settings changes).
3. **Reliability:** idempotent APIs, graceful failure, backups/seed scripts; no data loss on restart.
4. **Performance:** load test to **≥500 simulated voters** with acceptable latency; show results.
5. **Accessibility & UX:** responsive UI; clear flows; WCAG AA basics.

Data & API starter (you may refine)

Core tables/collections

- `Positions(id, name, seats=1, opens_at, closes_at)`
- `Candidates(id, position_id, name, manifesto_url, photo_url, status[SUBMITTED|APPROVED|REJECTED], reason, created_at)`
- `EligibleVoters(id, reg_no, name, email|phone, program, status[ELIGIBLE|BLOCKED])`
- `Verifications(id, voter_id, method, otp_hash, issued_at, verified_at, ballot_token, consumed_at)`
- `Ballots(id, voter_id, issued_at, consumed_at, election_id)`
- `Votes(id, ballot_id, position_id, candidate_id, cast_at)` *(no voter PII here)*
- `AuditLog(id, actor_type, actor_id, action, entity, entity_id, payload(json), created_at)`

Essential endpoints

- `POST /positions` (admin)
- `POST /candidates` (candidate) → `PATCH /candidates/:id/approve|reject` (officer)
- `POST /verify/request-otp` → `POST /verify/confirm` (issue single-use ballot token)
- `GET /ballot` (with token) → positions & candidates
- `POST /vote` (with token) → cast & **invalidate token**
- `GET /reports/turnout`, `GET /reports/results`, `GET /reports/audit`, `GET /export/:type`

Milestones & deliverables (20% each)

M1 — Discovery & Architecture

Tasks: stakeholder mapping, user journeys, wireframes for the 4 flows, ERD + sequence diagrams, risk/threat model (top 5 risks + mitigations), repo setup (issues/board/branching), CI lint/test.
Deliverables:

- PDF: brief architecture doc + wireframes + ERD + risk table
- Repo links (frontend & backend), CI badge, populated issue board.

M2 — Nominations Module

Tasks: positions CRUD, nomination form & upload, officer approval flow, status board, audit logging.
Deliverables:

- Working nominations UI + APIs
- Evidence of audit entries for submit/approve/reject
- 6–10 unit/integration tests touching validations & approvals.

M3 — Voter Verification & Ballot Setup

Tasks: import eligibility CSV, OTP flow, throttle/rate-limit, single-use ballot token, **no re-verification after vote**.
Deliverables:

- End-to-end verification to **ballot access**
- Negative tests (wrong OTP, expired token, blocked voter)
- Short screencast (≤2 min) demonstrating verification & token issuance.

M4 — Voting Engine & Live Dashboard

Tasks: secret ballot casting, replay/double-vote prevention, window enforcement, tally computation, live turnout widget, backup & restore script, **500-user load test** (report).
Deliverables:

- E2E voting works; double-vote blocked (show proof)
- Live dashboard (turnout + per-position counts)
- Load-test summary (method, numbers, graphs, findings).

M5 — Reporting & Final Presentation

Tasks: results & turnout reports (CSV/PDF), candidate list, audit export, admin SOP, deployment (web: GH Pages/Render; mobile: APK + demo).
Deliverables:

- Exports (CSV/PDF) for turnout, results, audit log
- 8–10 slide deck + 5–8 min demo video/live demo
- Concise **Admin SOP** (account recovery, closing election, exporting results)
- **Individual reflection** (≤300 words) on contributions & lessons.

S/N	Milestone Description	Maximum Marks
1	**MILESTONE ONE - Discovery & Architecture** Stakeholder mapping, user journeys, wireframes for 4 flows, ERD & sequence diagrams, risk model, CI/Git setup.	20 %
2	**MILESTONE TWO - Nominations Module** Positions CRUD, candidate nomination & uploads, officer approval/rejection, status board, audit logging.	20 %
3	**MILESTONE THREE - Voter Verification & Ballot Setup** Eligibility CSV import, OTP verification flow, rate-limiting, single-use ballot token, block re-verification after vote.	20 %
4	**MILESTONE FOUR - Voting Engine & Live Dashboard** Secret ballot casting, double-vote/replay prevention, window enforcement, live turnout & tally dashboard, load test (≥500)	20 %
5	**MILESTONE FIVE - Reporting & Final Presentation** Results & turnout reports (CSV/PDF), audit export, admin SOP, deployment + demo, individual reflections.	20 %
	TOTAL MARKS	**100 %**

PART 2 : Marking & individualization

1. **Team mark per milestone (0–20).**
2. **Individual score** = Team mark × **Contribution Factor (0.6–1.1)** from:
   a. Commits/PRs/reviews linked to assigned issues
   b. Test ownership/coverage
   c. Issue tracker participation & delivery reliability
   d. Clarity of individual reflection at M5

Technology & delivery constraints

- GitHub (two repos), **feature branches + PR reviews** required.
- Add a `/docs` folder for diagrams & SOP.
- .env.example with all required secrets/URLs; never commit real secrets.
- Provide a **seed script** to create sample positions, 10–20 candidates, and 1000 eligible voters.
- For SMS/email OTP, you may **mock** delivery in development (console/email preview).

CSV schemas to use in testing

- **eligible_voters.csv**: `reg_no,name,email,phone,program,status`
- **candidates_seed.csv**: `position,name,manifesto_url,photo_url`
- **positions.csv**: `name,seats,opens_at,closes_at`

Integrity & policy notes

- Follow university exam rules and academic integrity.
- All third-party code must be attributed; no copy-pasting full solutions.
- Deadlines are firm; late submissions are not accepted.

Submission checklist

- Frontend & backend repo links (with CI)
- Deployed link or APK + demo video
- Architecture PDF + ERD + risk model
- Evidence of audit logs (screenshots/JSON sample)
- Load-test report (≥500 users)
- CSV/PDF exports (turnout, results, audit)
- Slides + individual reflections

Important Notes:

- **Violation of Policies:** Any violation of Uganda Christian University's examination guidelines or academic policies will result in the examination attempt being void, regardless of completion.
- **Deadline:** The project must be completed and submitted within **two weeks.** Late submissions will not be accepted.
- **Plagiarism:** Plagiarism or any form of academic dishonesty will result in severe penalties, including the possibility of failing the exam.

This practical exam is designed to test your ability to work collaboratively, plan, develop, and present a functional web or mobile application. Good luck!

~END OF EXAM GUIDELINES~