Postgraduate Request Portal – Functional Specification

1. Purpose of This Document

This document defines the **functional requirements** and **workflow logic** for the Postgraduate Request Portal. It is intended as a **handover and briefing document for a software developer intern** who will assist in completing the portal to a professional, production-ready standard.

The portal's primary goal is to **digitise, track, and govern Higher Degree (HD) administrative processes** while ensuring transparency, accountability, and role-based control across the postgraduate lifecycle.

---

2. User Roles and Access Control

The system supports **four user roles**, each with clearly defined permissions. Role-based access control (RBAC) is mandatory.

2.1 Student

Students are the initiators of Higher Degree (HD) requests and the owners of their academic administrative records.

Core Student Functions

2.1.1 Higher Degree Requests (HD Requests)

- Complete HD requests using **embedded JotForms** (external form logic is preserved).
- Submit requests into the portal workflow.
- View request status in real time.

2.1.2 Submission Tracking (Current Requests)

- Track each active submission through the HD committee cycle:
    - Draft
    - Submitted to Supervisor
    - Supervisor Review

    - Co-Supervisor Review (if applicable)
    - Coordinator Review
    - Faculty Higher Degrees (FHD)
    - Senate Higher Degrees (SHD)
    - Approved / Recommended / Referred Back
- See timestamps and current owner of the request.

2.1.3 Academic Progress Tracker (Historical Overview)

- View:
    - Years registered
    - Degree programme
    - Supervisor history
    - All completed HD requests
    - Outcomes and reference numbers
- Requests that have completed the HD cycle become **read-only historical records**.

2.1.4 Year Calendar & Deadlines

- View a year-based calendar containing:
    - HD submission deadlines
    - Committee meeting dates
    - Faculty events
    - Automated reminders for stagnant submissions

2.1.5 Milestones & Professional Development

- Add non-HD milestones, such as:
    - Conference presentations / attendance
    - Journal clubs
    - Training courses and workshops
- Milestones are visible to supervisors and coordinators.

2.2 Supervisor

Supervisors are reviewers and mentors responsible for guiding students through HD processes.

Core Supervisor Functions

2.2.1 Student Progress Monitoring

- View a dashboard of all assigned students.
- Track:
    - Active submissions

    - Historical submissions
    - Academic progress summaries

2.2.2 HD Request Review & Approval

- Receive HD requests via **secure access code workflow** (see Section 4).
- Edit requests where permitted.
- Digitally sign requests.

2.2.3 Student Engagement Tools

- “Nudge” students via in-system notifications to:
    - Submit pending requests
    - Respond to edits
    - Attend events or meet deadlines

2.3 Coordinator (Postgraduate Coordinator)

The coordinator oversees HD processes at programme or faculty level.

Core Coordinator Functions

2.3.1 Global Student Oversight

- View all postgraduate students under their jurisdiction.
- Track:
    - Current submissions
    - HD pipeline status
    - Aggregate progress metrics

2.3.2 HD Committee Preparation

- Export **spreadsheet summaries** for HD committee meetings including:
    - Student name & number
    - Degree
    - Supervisor(s)
    - Request type
    - Submission date to HD secretary
    - Current status

2.3.3 Student Record Management

- Update student records:
    - Thesis title changes

    - Add/remove supervisors
    - Correct metadata

2.3.4 Progress Tracker Updates

- Update HD outcomes:
    - FHD: Approved / Recommended / Referred Back
    - SHD: Approved / Referred Back
- Capture HD reference numbers.

2.3.5 Calendar Management

- Create and update year calendar events and reminders.

2.4 Portal Administrator

The administrator manages governance, data integrity, and system oversight.

Core Administrator Functions

2.4.1 Role & Permission Management

- Assign and update user roles.
- Override access where required.

2.4.2 Data Export & Reporting

- Export:
    - Student datasets
    - Supervisor datasets
    - Submission logs

2.4.3 Notification & Reminder Engine

- Receive automated alerts at every submission stage.
- Monitor stalled workflows.

2.4.4 Calendar & Scheduling

- Manage institution-wide events and reminders.

2.4.5 Document Repository Access

- Access **Google Drive repository** containing final, signed PDFs.
- Ensure correct folder structure and permissions.

2.4.6 Audit Logs & Analytics

- View:
    - User activity logs
    - Submission timelines
    - System usage analytics

---

3. Core System Modules

3.1 Authentication & Identity

- Secure login (email-based).
- Role-based dashboards.
- Password reset & account recovery.

3.2 Submission Tracker Engine

- State-machine–based workflow.
- Every submission exists in **one and only one state at any time**.
- State transitions are logged and auditable.

3.3 Notification System

- In-app notifications.
- Email notifications.
- Time-triggered reminders (e.g. stagnation alerts).

3.4 Calendar Module

- Role-aware visibility.
- Sync-ready design (future integration).

---

4. HD Request Workflow Logic (Critical)

4.1 Access Code–Based Review Mechanism

Generation & Rules

- Each review step generates a **unique, random access code**.

- Access codes:
    - Are visible to sender and recipient only.
    - Expire after **72 hours**, unless otherwise specified.

4.2 Student → Supervisor Workflow

1. Student submits HD request.
2. System generates access code and sends to supervisor.
3. Supervisor has **48 hours** from first open to:
    - Review
    - Edit
    - Save changes
4. If edits are saved:
    - Student is notified.
    - Student has **24 hours** to approve edits.
5. Once approved:
    - Supervisor signs digitally.

4.3 Supervisor → Co-Supervisor Workflow

- Triggered only if co-supervisor exists.
- New, different access code generated.
- Co-supervisor:
    - Cannot edit
    - Can review and sign
    - Can participate in chat

Chat Logic

- Google Chat becomes active when document is opened.
- While chat is active:
    - Access code timer is suspended.

4.4 Multi–Co-Supervisor Handling

- Sequential signing.
- New access code generated per co-supervisor.
- Document locks only after final signature.

4.5 Coordinator Finalisation

- Coordinator receives final signed document.
- Coordinator signs.
- System:
    - Locks document
    - Generates final PDF
    - Uploads to Google Drive
    - Notifies administrator

---

5. HD Committee Decision Logic

5.1 Faculty Higher Degrees (FHD)

Coordinator records:

- Reference number
- Outcome:
    - Approved
    - Recommended
    - Referred Back

5.2 Senate Higher Degrees (SHD)

- If FHD = Approved → SHD auto-checked.
- If FHD = Recommended → SHD field activated.

5.3 Referred Back Handling

- Coordinator records reason.
- Supervisor notified.
- Supervisor has **24 hours** to amend and resubmit.

---

6. Suggested Additional Logic Enhancements

- **Versioning:** Maintain immutable versions of each submission.
- **Escalation Rules:** Auto-escalate stalled submissions.
- **Conflict Detection:** Flag inconsistent edits.
- **Visual Workflow Map:** Show submission journey visually.

---

7. Non-Functional Requirements

- Security: POPIA-compliant
- Auditability: Full trace logs
- Availability: 99% uptime target
- Maintainability: Modular codebase

---

8. Intended Outcome

This portal should function as a **transparent, auditable, and user-friendly governance system** for postgraduate administration, reducing email dependency, preventing lost documents, and providing real-time visibility for all stakeholders.

This document should be treated as the **authoritative functional reference** for further development.

Additional insights

This is a solid, non-trivial system you're specifying — the kind that benefits hugely from having expectations written down clearly before a developer starts "just building features".
I've created a full functional specification document in the canvas that you can hand directly to the intern. It's written to be:
Role-based and unambiguous (so they don't guess who can do what)
Workflow-driven (state changes, access codes, signatures, timers)
Implementation-aware but not code-specific (so it works regardless of stack)
Governance-heavy (audit logs, locking, traceability — which your portal really needs)
How I suggest you use this
Give it to the intern as required reading before coding
Ask them to:
Identify missing edge cases
Propose a data model (tables/entities) from it
Sketch a state diagram for the HD workflow
Use their response to assess whether they're thinking like a system builder, not just a coder
If you want, next we can
Turn this into a developer task backlog (epics → user stories)

Extract a clean ERD / database schema
Define API endpoints per role
Stress-test the workflow for failure cases (expired codes, silent supervisors, conflicting edits)