UWC PostGrad Portal

README — Project Documentation

University of the Western Cape

Generated: 8 February 2026

Project: Postgraduate Request Portal

Methodology: Design Science Research (DSRM)

Table of Contents

UWC PostGrad Portal

A role-based postgraduate request management system for the University of the Western Cape (UWC). Built with React 19, Vite, Firebase (Authentication + Cloud Firestore), and standard CSS.

Features

Role	Capabilities
Student	Create/submit HD requests, track submissions, manage milestones, view academic progress
Supervisor	Review requests, approve/refer back, sign digitally, nudge students, annotate documents
Coordinator	Manage all requests, forward to Faculty/Senate Boards, record outcomes, export agendas
Admin	System overview, analytics, role management, audit logs, dataset exports

* HD Request Workflow -- Full lifecycle: draft -> supervisor review -> co-supervisor sign -> coordinator -> Faculty Board -> Senate Board -> approved
* Access Code System -- Secure supervisor access via generated 6-character codes with expiry
* Digital Signatures -- Draw or type signature pad for approval actions
* Submission Tracker -- Visual workflow progress bar with owner tracking and response timers
* Document Review & Version Control -- Multi-version document management with visual diff, feedback, and status workflow (submitted -> under review -> changes requested -> approved)
* PDF Annotation System -- Full-screen PDF viewer (react-pdf) with text selection, inline highlight annotations, colour picker, reply threads, resolve/reopen, and draft -> sent batch workflow
* Batch Annotation Workflow -- Supervisors/coordinators save annotations as drafts, review all at once, then confirm & send to notify the student via in-app notification + email
* Nudge System -- Supervisors can send reminder notifications to students
* Committee Exports -- CSV export of Faculty/Senate Board agendas with student number, degree, and supervisor
* Refer-Back Workflow -- 24-hour amendment timer when requests are referred back
* Calendar -- Full CRUD calendar with month view, role-based auto-filtering, and event types
* Milestones -- Students can log academic milestones (conferences, publications, etc.)

* Audit Logs -- Searchable activity log with date filtering and CSV export
* Analytics Dashboard -- Bar charts for request status/type distribution, summary statistics
* Role Management -- Admin interface to reassign user roles
* Overdue Monitoring -- "Overdue Only" filter on requests page for coordinators/admins
* Notifications -- Real-time Firestore-backed notifications with bell icon, unread count, mark-read, and link navigation
* Email Notifications -- EmailJS integration for real email delivery on all cross-user actions (submissions, approvals, annotations, comments, feedback, nudges)
* Dark/Light Mode -- Complete theming system with persistent preference via Settings -> Appearance tab. 100+ element-level CSS overrides across all pages ensure consistent styling in both themes
* Help & Documentation -- Dedicated Help & Docs page (all roles) with 7 FAQ categories (20+ questions), 6 static guides, and 13 interactive walkthrough tours
* Guided Tour System -- Overlay-based walkthrough engine with SVG mask highlighting, auto-scroll to elements, click-to-proceed steps, cross-page navigation, and 4 role-specific full system tours

Four User Roles

Core Functionality

Tech Stack

Install dependencies

npm install

Configure environment variables

cp .env.example .env

Edit .env with your Firebase and EmailJS credentials (see below)

Start development server

npm run dev

Production build

npm run build

Preview production build

npm run preview

Create a .env file in the project root:

VITE_FIREBASE_API_KEY=your-api-key
VITE_FIREBASE_AUTH_DOMAIN=your-project.firebaseapp.com
VITE_FIREBASE_PROJECT_ID=your-project-id
VITE_FIREBASE_STORAGE_BUCKET=your-project.firebasestorage.app
VITE_FIREBASE_MESSAGING_SENDER_ID=your-sender-id
VITE_FIREBASE_APP_ID=your-app-id

VITE_EMAILJS_SERVICE_ID=your-emailjs-service-id
VITE_EMAILJS_TEMPLATE_ID=your-emailjs-template-id
VITE_EMAILJS_PUBLIC_KEY=your-emailjs-public-key

Email	Role	Name
student@uwc.ac.za	Student	Thabo Molefe
student2@uwc.ac.za	Student	Amahle Dlamini
student3@uwc.ac.za	Student	Sipho Dlamini
supervisor@uwc.ac.za	Supervisor	Prof. Sarah van der Berg
supervisor2@uwc.ac.za	Supervisor	Dr. Marcus Thompson
coordinator@uwc.ac.za	Coordinator	Dr. Fatima Patel
admin@uwc.ac.za	Admin	Linda Mkhize

> *Default password: Portal@2026*

Environment Variables

Demo Accounts

Project Structure

postgrad-portal/
--- index.html # Entry HTML (UWC favicon)
--- package.json
--- firebase.json # Firebase project config
--- firestore.rules # Firestore security rules
--- vite.config.js
--- public/
- --- uwc_logo.svg # University of the Western Cape logo
- --- documents/ # 19 generated sample documents (PDF/DOCX/XLSX)
- --- hdr-001/ # Progress_Report, Publication_Evidence, Supervisor_Feedback
- --- hdr-002/ # Research_Proposal, Literature_Review, Ethics_Clearance
- --- hdr-003/ # Extension_Motivation_Letter
- --- hdr-004/ # Ethics_Application, Informed_Consent, Data_Collection
- --- hdr-005/ # Registration, Academic_Transcript, Progress_Report
- --- hdr-006/ # Examiner_Nomination, Turnitin_Report
- --- hdr-007/ # Progress_Report_NK
- --- hdr-009/ # Medical_Certificate
- --- hdr-011/ # Title_Registration_Form
- --- hdr-012/ # Progress_Report_Jan2026
--- scripts/ # Automation & provisioning
- --- setup-firebase.mjs # Firebase Auth user creation
- --- seed-firebase.mjs # Firestore collection seeding (7 collections)
- --- reseed-firebase.mjs # Clear & re-seed all data
- --- seed-document-versions.mjs # Document version history seeding
- --- seed-annotations.mjs # Annotation seed data
- --- upload-documents.mjs # Generate 13 sample PDFs/DOCX/XLSX with pdf-lib
- --- generate-missing-pdfs.mjs # Generate 6 additional PDFs referenced by versions
- --- test-emailjs.mjs # EmailJS connectivity test
- --- test-firestore.mjs # Firestore connectivity test
--- docs/
- --- FIREBASE_MIGRATION_CHANGELOG.md
- --- EMAILJS_SETUP.md
--- src/
--- App.jsx # Routes and protected route wrapper
--- main.jsx # Entry point (AuthProvider + DataProvider)
--- index.css # Global styles & CSS custom properties
--- components/
- --- common/ # Shared UI - Card, Modal, StatusBadge, Avatar, etc.
- - --- index.jsx
- - --- common.css
- - --- SignaturePad.jsx

- --- layout/ # App shell - sidebar, header, outlet
- - --- Layout.jsx
- - --- Header.jsx # Notification bell dropdown
- - --- Sidebar.jsx # UWC-branded nav with role-specific menus
- - --- layout.css
- --- AnnotatedDocViewer.jsx # Full-screen PDF viewer + annotation system
- --- AnnotatedDocViewer.css
--- context/
- --- AuthContext.jsx # Firebase Auth provider
- --- DataContext.jsx # Firestore real-time subscriptions + mutations
- --- ThemeContext.jsx # Dark/light mode provider (localStorage persistence)
- --- GuidedTour.jsx # Guided tour engine (overlay, highlight, auto-scroll)
- --- GuidedTour.css # Tour overlay + tooltip styles
--- firebase/
- --- config.js # Firebase app initialisation
- --- firestore.js # Firestore CRUD (all collections)
- --- collections.js # Collection name constants
- --- documentVersions.js # Document version control operations
- --- annotations.js # Annotation CRUD + batch confirm/send
- --- storage.js # Firebase Storage helpers
--- services/
- --- emailService.js # EmailJS integration (send real emails)
- --- pdfService.js # Client-side PDF generation (jsPDF)
--- pages/
- --- Dashboard.jsx # Role-based dashboard router
- --- HDRequestsPage.jsx # Request list, detail modal, all workflow actions
- --- DocumentReviewPage.jsx # Version control, comments, feedback, annotations
- --- SubmissionTracker.jsx # Visual workflow progress tracker
- --- CalendarPage.jsx # Month calendar with CRUD
- --- StudentsPage.jsx # Student directory with edit modal
- --- AcademicProgressPage.jsx # Student academic history
- --- AnalyticsPage.jsx # Admin analytics with charts
- --- AuditLogsPage.jsx # Searchable audit log
- --- RoleManagementPage.jsx # Admin role assignment
- --- LoginPage.jsx # Login with demo quick-access
- --- SettingsPage.jsx # Profile, notifications, password, appearance (theme)
- --- HelpPage.jsx # Help & Docs (FAQs, guides, interactive walkthroughs)
- --- HelpPage.css # Help page styles
- --- walkthroughs.js # 13 walkthrough definitions (4 full + 9 task-specific)
- --- SeedPage.jsx # Admin-only database reseed tool
- --- dashboards/
- --- StudentDashboard.jsx
- --- SupervisorDashboard.jsx
- --- CoordinatorDashboard.jsx

- --- AdminDashboard.jsx
--- utils/
--- constants.js # Status configs, labels, workflow states
--- helpers.js # Date formatting, utilities

Routes

System vs Specification

Next Section

UWC PostGrad Portal

System vs Specification — Comparison & Analysis

University of the Western Cape

Generated: 8 February 2026

Project: Postgraduate Request Portal

Methodology: Design Science Research (DSRM)

Table of Contents

System vs Functional Specification - Comparison & Analysis

> *Project: UWC Postgraduate Request Portal*
> *Reference Document: Postgraduate Request Portal - Functional Specification*
> *System Build Date: February 2026*
> *Methodology: Design Science Research Methodology (DSRM)*

Table of Contents

1. Purpose

This document provides a systematic comparison between the original Functional Specification and the implemented PostGrad Portal system. It identifies:

* Fully implemented features that match or exceed the specification
* Partially implemented features with noted deviations
* Not implemented features with rationale for omission
* Beyond-spec features that were added based on usability testing or design science iteration findings

This analysis supports the DSRM evaluation phase by demonstrating traceability between requirements

and artefact.

Development Changelog

Next Section

UWC PostGrad Portal

Development Changelog & Research Notes

University of the Western Cape

Generated: 8 February 2026

Project: Postgraduate Request Portal

Methodology: Design Science Research (DSRM)

Table of Contents

PostGrad Portal - Development Changelog & Research Notes

> *Project: Postgraduate Request Portal*
> *Context: Academic Research Project (Design Science Research Methodology)*
> *Date: February 2026*
> *Scope: Document Review System, Annotation Engine, Notification Architecture, and Codebase Cleanup*

Table of Contents

1. Overview

This document records the second major development iteration of the PostGrad Portal, building upon the Firebase migration documented in FIREBASE_MIGRATION_CHANGELOG.md. This iteration focused on:

* Document Review & Version Control -- a complete multi-version document review system with comments, feedback, and status workflow
* PDF Annotation Engine -- full-screen PDF viewer with text-selection-based annotations, highlight overlays, reply threads, and resolve/reopen functionality
* Batch Annotation Workflow -- draft -> confirm -> send pattern for supervisors to batch-review annotations before sending to students
* Comprehensive Notification Architecture -- replacing broken mock in-memory notifications with Firestore-persisted notifications + EmailJS email delivery across all cross-user actions
* Document Generation -- 19 realistic sample documents (PDF, DOCX, XLSX) generated with pdf-lib,

docx, and exceljs for end-to-end testing
* UWC Branding -- official SVG logo integration across sidebar, login page, and browser favicon
* Codebase Cleanup -- removal of the legacy mockData.js (627 lines) and all dead code dependencies

This work constitutes the third DSRM iteration of the artefact:

Iteration 4 -- Help System, Guided Tours, Dark/Light Mode & Specification Comparison

> *Date: February 2026*
> *Scope: Help & Documentation page, interactive guided tours, theme system, comprehensive dark mode, system vs specification analysis*

14. Dark/Light Mode (Theme System)

The portal had a single light theme with no user preference support. Extended use by academics (supervisors reviewing documents, coordinators processing queues) required reduced eye strain options.

A dedicated ThemeContext.jsx provides:

* State: theme (string: 'light' | 'dark'), setTheme(t), toggleTheme()
* Persistence: localStorage key pgportal-theme
* DOM integration: Sets data-theme attribute on document.documentElement, enabling CSS cascade
* Provider: Wraps the app in App.jsx above the Router

1. CSS custom properties in :root define the light theme (default)
2. A [data-theme="dark"] selector block overrides all 40+ custom properties (backgrounds, borders, text, shadows, status colours)
3. Element-level overrides (100+ selectors) handle elements that use hardcoded colours, inline styles, or component-specific implementations that can't rely on variables alone

Categories of dark overrides:
* Global: body, scrollbars, sidebar, header, search
* Forms: inputs, selects, textareas, placeholders, focus rings
* Tables: th, td, zebra striping
* Modals: overlay backdrop, modal card
* Components: filter chips, tabs, notification dropdown, empty states
* Pages: calendar, toast, login form, annotation viewer, signature pad, file upload zone

SettingsPage.jsx now has an Appearance tab (alongside Profile, Notifications, Security) with:

* Two visual cards (Light Mode with sun icon, Dark Mode with moon icon)
* Active state ring on the currently selected theme
* Instant theme apply on click (no save button needed)

14.1 Problem Statement

14.2 Solution: ThemeContext

14.3 CSS Implementation Strategy

14.4 Settings Integration

15. Guided Tour Engine

The portal serves four user roles with distinct workflows. New users (particularly students joining each academic year) need onboarding without external documentation.

A context-based overlay system (GuidedTour.jsx, ~250 lines) providing:

Core Architecture:
TourProvider (wraps Layout)
--- State: activeTour, stepIndex, highlight rect, tooltip position
--- positionStep()
- --- Find DOM element by CSS selector
- --- scrollIntoView({ behavior: 'smooth', block: 'center' })
- --- Wait 350ms for scroll to settle
- --- Compute bounding rect with padding
- --- Calculate tooltip position (bottom/top/left/right/center)
--- Route navigation (useNavigate) for cross-page tours
--- Click-to-proceed: attach click listener to target element
--- Window resize: reposition on viewport change
--- TourOverlay (portal to document.body)
--- SVG mask backdrop with rectangular cutout
--- Highlight ring (gold pulse animation)
--- Tooltip card (title, content, step counter)
--- Progress dots (active/completed states)
--- Navigation buttons (Previous, Next, End Tour)

Key Design Decisions:

* SVG mask cutout -- a full-viewport SVG <rect> with a <mask> exclusion creates a dark backdrop with a transparent window over the highlighted element. This approach works regardless of the element's

z-index or stacking context.
* Fixed dark tooltip -- the tooltip uses a dark background (#1a2332) with white text and gold (#C5A55A) accents, ensuring readability in both light and dark page themes.
* Route-aware steps -- each step can specify a route property. If the current page differs, the engine navigates first, then waits 500ms for the DOM to settle before positioning the highlight.

walkthroughs.js (~350 lines) defines 13 walkthroughs grouped into categories:

Full System Tours (per role):

Task-Specific Tours:

Helper function: getToursForRole(role) filters tours to only show those relevant to the user's role.

Firebase Migration Changelog

Next Section

UWC PostGrad Portal

Firebase Migration Changelog

University of the Western Cape

Generated: 8 February 2026

Project: Postgraduate Request Portal

Methodology: Design Science Research (DSRM)

Table of Contents

PostGrad Portal - Firebase Migration Changelog

> *Project: Postgraduate Request Portal*
> *Context: Academic Research Project (Design Science Research Methodology)*
> *Date: June 2025*
> *Scope: Migration from in-memory mock data layer to Firebase (Authentication + Cloud Firestore)*

Table of Contents

1. Overview

The PostGrad Portal is a React-based university postgraduate administration system supporting four user roles: Student, Supervisor, Coordinator, and Admin. Prior to this migration, the application relied entirely on an in-memory mock data layer (mockData.js, ~627 lines) that simulated CRUD operations with JavaScript arrays and objects. Data was volatile -- lost on every page refresh.

This changelog documents the complete migration from mock data to Firebase, introducing:

* Firebase Authentication (email/password) replacing simulated login
* Cloud Firestore replacing the in-memory data store with persistent, real-time data
* Real-time subscriptions (onSnapshot) so all connected clients receive live updates
* Automation scripts for repeatable provisioning and seeding

This migration constitutes a core design science artefact iteration -- transforming a functional prototype into a production-capable system while preserving all existing UI behaviour.

EmailJS Setup Guide

Next Section

UWC PostGrad Portal

EmailJS Setup Guide

University of the Western Cape

Generated: 8 February 2026

Project: Postgraduate Request Portal

Methodology: Design Science Research (DSRM)

Table of Contents

EmailJS Setup Guide -- PostGrad Portal

This guide walks you through connecting EmailJS so the Portal can send email notifications (submission alerts, approval notices, deadline reminders, etc.).

Free tier: 200 emails / month, 2 templates. Enough for testing & small departments.

1. Connect an Email Service

1. Open <https://dashboard.emailjs.com/admin>
2. Click Email Services -> Add New Service.
3. Choose your provider:

* Gmail (easiest for testing) - click Connect Account, sign in with a Google account, grant permission.
* Or choose Outlook / SMTP / etc.

4. Name the service something like postgrad_portal.
5. Click Create Service.
6. Copy the Service ID shown (e.g. service_abc1234) -> you'll need it in step 3 below.

2. Create a Notification Template

1. In the dashboard go to Email Templates -> Create New Template.
2. Set Template Name to portal_notification.
3. Paste the following content into the template editor:

```
{{subject}}

<div style="font-family:Arial,sans-serif;max-width:600px;margin:0 auto;">
<h2 style="color:#2563eb;">PostGrad Portal</h2>
<p>Dear {{to_name}},</p>
<div style="white-space:pre-line;">{{message}}</div>
<br/>
<a href="{{action_url}}"
style="display:inline-block;padding:12px 24px;background:#2563eb;color:#fff;
text-decoration:none;border-radius:6px;">
{{action_text}}
</a>
```

```
<hr style="margin-top:32px;border:none;border-top:1px solid #e5e7eb;"/>
<p style="font-size:12px;color:#6b7280;">
This email was sent by {{from_name}}. If you did not expect this message,
please ignore it.
</p>
</div>
```

4. In the To Email field, set it to {{to_email}}.
5. Set From Name to {{from_name}}.
6. Click Save.
7. Copy the Template ID (e.g. template_xyz789).