

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	3
Table of Contents	3
1. Overview	3
Iteration 4 -- Help System, Guided Tours, Dark/Light Mode & Specification Compar...	4
14. Dark/Light Mode (Theme System)	4
14.1 Problem Statement	5
14.2 Solution: ThemeContext	5
14.3 CSS Implementation Strategy	5
14.4 Settings Integration	5
15. Guided Tour Engine	5

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
2. Document Review & Version Control System
3. PDF Annotation Engine
4. Batch Annotation Workflow
5. Notification Architecture
6. Email Integration (EmailJS)
7. Document Generation & Management
8. UWC Branding & Logo Integration
9. Codebase Cleanup - Mock Data Removal
10. Challenges Faced
11. Lessons Learned
12. Design Science Alignment
13. File Inventory

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.