# Software Requirements Specification Online Course Registration and Enrollment System

Version 1.0 – Final

Prepared by Carlon Bullock

CST499: Capstone for Computer Software Technology

Professor: John Russell

April 14, 2025

# Table of Contents

[Software Requirements Specification Online Course Registration and Enrollment System 1](#_Toc195550294)

[Table of Contents 2](#_Toc195550295)

[Revision History 3](#_Toc195550296)

[1  Introduction 4](#_Toc195550297)

[1.1 Purpose 4](#_Toc195550298)

[1.2 Document Conventions 4](#_Toc195550299)

[1.3 Intended Audience and Reading Suggestions 4](#_Toc195550300)

[1.4 Project Scope 4](#_Toc195550301)

[1.5 References 4](#_Toc195550302)

[2  Overall Description 5](#_Toc195550303)

[2.1 Product Perspective 5](#_Toc195550304)

[2.2 Product Features (summary) 5](#_Toc195550305)

[2.3 User Classes & Characteristics 5](#_Toc195550306)

[2.4 Operating Environment 5](#_Toc195550307)

[2.5 Design & Implementation Constraints 5](#_Toc195550308)

[2.6 User Documentation 6](#_Toc195550309)

[2.7 Assumptions & Dependencies 6](#_Toc195550310)

[3  System Features 6](#_Toc195550311)

[3.1  F1 – User Registration 6](#_Toc195550312)

[3.2  F2 – Authentication & Authorization 6](#_Toc195550313)

[3.3  F3 – Course Catalog Browsing 7](#_Toc195550314)

[3.4  F4 – Enrollment Management 7](#_Toc195550315)

[3.5  F5 – Wait‑list Automation 7](#_Toc195550316)

[3.6  F6 – Enrollment Cancellation 7](#_Toc195550317)

[3.7  F7 – Profile Management 7](#_Toc195550318)

[4  External Interface Requirements 7](#_Toc195550319)

[4.1 User Interfaces 7](#_Toc195550320)

[4.2 Hardware Interfaces 7](#_Toc195550321)

[4.3 Software Interfaces 7](#_Toc195550322)

[4.4 Communications Interfaces 8](#_Toc195550323)

[5  Non-Functional Requirements 8](#_Toc195550324)

[5.1 Performance 8](#_Toc195550325)

[5.2 Safety 8](#_Toc195550326)

[5.3 Security 8](#_Toc195550327)

[5.4 Software Quality Attributes 8](#_Toc195550328)

[6  Other Requirements 8](#_Toc195550329)

[7  References 9](#_Toc195550330)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Carlon Bullock | 4/14/25 | Initial Approved Release | 1.0 |
|  |  |  |  |

# 1  Introduction

## 1.1 Purpose

This Software Requirements Specification (SRS) defines the functional and non-functional requirements for the **Online Course Registration and Enrollment System (OCRES)**. This web-based application enables students to register for semester-based courses, manage wait‑lists, and maintain personal profiles. The document serves as the contractual baseline between stakeholders and the development team.

## 1.2 Document Conventions

* **Shall** = mandatory requirement.
* **Should** = desirable requirement.
* Requirements are uniquely tagged REQ‑F#.# (functional) or REQ‑NF#.# (non-functional).
* IEEE 830 structure is followed.

## 1.3 Intended Audience and Reading Suggestions

* **Developers & QA** – Sections 2–5.
* **Project & Product Managers** – Sections 1–3, 5.
* **Security & IT Ops** – Sections 4–5.
* **Accreditation / Compliance** – Section 5.

## 1.4 Project Scope

OCRES delivers self-service registration across three academic terms (spring, summer, and fall). Students may browse the catalog, enroll if seats are available, join a wait-list if not, cancel enrollments, and receive automated notifications. Administrative staff configure course capacities and schedules. The system will be built iteratively using a hybrid Scrum/Kanban process (Gemino, Reich, & Serrador, 2020) that emphasizes short, potentially shippable increments and continuous stakeholder feedback (Tsui, Karam, & Bernal, 2022, Ch. 5).

## 1.5 References

The complete reference list appears in Section 7. Inline citations use APA author-date style.

# 2  Overall Description

## 2.1 Product Perspective

OCRES is a **new subsystem** that interfaces with the university Student Information System (SIS) and Learning Management System (LMS) via REST APIs. It will be deployed on the institution’s cloud Kubernetes cluster. The life‑cycle follows an **incremental, risk-driven spiral inside time-boxed sprints** (Tsui et al., 2022, Ch. 4).

## 2.2 Product Features (summary)

1. Account creation & profile management.
2. Secure authentication (SSO) and authorization.
3. Course catalog browsing with real-time seat counts.
4. Enrollment & wait-listing with FIFO promotion.
5. Self-service cancellation with automated notifications.
6. Admin dashboards for course setup and capacity tuning.

## 2.3 User Classes & Characteristics

| **Class** | **Description** | **Tech skill** |
| --- | --- | --- |
| Applicant | Prospective student registering for the first time | Low |
| Student | Authenticated user managing enrollments | Medium |
| Registrar Staff | Configure courses, capacities, dates | High |
| System Admin | Maintain servers, security, and backups | High |

## 2.4 Operating Environment

* Client: Modern browsers (Chrome, Edge, Firefox, Safari).
* Server: Ubuntu 22.04 LTS, Node 20/Express, PostgreSQL 15, Nginx reverse proxy.
* Protocols: HTTPS (TLS 1.3), OAuth 2.1/OIDC.

## 2.5 Design & Implementation Constraints

* Must comply with FERPA privacy rules.
* Must integrate with campus SSO.
* Continuous integration pipeline with GitHub Actions and automated unit/integration tests per sprint (Tsui et al., 2022, Ch. 5).

## 2.6 User Documentation

* In-app contextual help.
* PDF quick‑start guide.
* 3-minute video tutorials.

## 2.7 Assumptions & Dependencies

* SIS & LMS APIs remain stable.
* Users possess valid email for verification.
* Development team capacity = 6 FTE (XP‑style small team per Beck, as cited in Tsui et al., 2022, Ch. 5).

# 3  System Features

Each feature lists a description, stimulus/response, and verifiable requirements.

## 3.1  F1 – User Registration

**Description & Priority** (H) Applicants create unique IDs and profiles.  
**Stimulus/Response**: Applicant submits form → system validates uniqueness → sends verification email → activates account.  
**Functional Requirements**

* REQ‑F1.1 The system **shall** require a unique userID, hashed password, legal name, phone, and email.
* REQ‑F1.2 The system **shall** reject duplicate userID or emails and display an explanatory error.
* REQ‑F1.3 The system **shall** hash passwords using Argon2id with 16-byte salt.
* REQ‑F1.4 The system **shall** send a one-time verification link that expires in 24 hours.

## 3.2  F2 – Authentication & Authorization

* REQ‑F2.1 The system **shall** authenticate via campus SSO or local credentials.
* REQ‑F2.2 After five failed attempts within 15 minutes, the account **shall** be locked for 30 minutes.

## 3.3  F3 – Course Catalog Browsing

* REQ‑F3.1 Users **shall** filter courses by semester, department, and keyword.
* REQ‑F3.2 System **shall** display remaining seats and wait-list length in real-time.

## 3.4  F4 – Enrollment Management

* REQ‑F4.1 System **shall** enroll students if seatsAvailable > 0.
* REQ‑F4.2 If full, the system **shall** append the student to the wait‑list FIFO.
* REQ‑F4.3 System **shall** enforce maxEnrollment configured per course.

## 3.5  F5 – Wait‑list Automation

* REQ‑F5.1 When a seat is free, the system **shall** notify the first wait-listed student via email and in-app message.
* REQ‑F5.2 Students **shall** have 24 hours to accept; otherwise, the next student will be notified.

## 3.6  F6 – Enrollment Cancellation

* REQ‑F6.1 Students **shall** cancel any enrollment before the add/drop deadline.
* REQ‑F6.2 Cancellation **shall** trigger wait-list promotion per REQ‑F5.1.

## 3.7  F7 – Profile Management

* REQ‑F7.1 Users **shall** update phone/email; email change requires re-verification.

# 4  External Interface Requirements

## 4.1 User Interfaces

Responsive SPA was built with React 18 and is WCAG 2.2 AA compliant. Wireframes will be reviewed with users each sprint (Tsui et al., 2022, Ch. 6).

## 4.2 Hardware Interfaces

None; cloud‑hosted.

## 4.3 Software Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Direction | Protocol | Purpose |
| SIS API v3 | bi‑dir | REST/JSON | Verify student status; write enrollments |
| LMS API v2 | out | REST/JSON | Push roster changes |
| SMTP | out | RFC 5321 | Send notifications |

## 4.4 Communications Interfaces

All traffic over HTTPS TLS 1.3; OAuth 2.1 tokens for API calls.

# 5  Non‑Functional Requirements

## 5.1 Performance

* REQ‑NF1.1 System **shall** support 500 concurrent users with < 2 s median page load.
* REQ‑NF1.2 99th‑percentile enrollment transaction latency ≤ 1 s.

## 5.2 Safety

Weekly encrypted backups are stored across‑regions; a quarterly restore drill is performed.

## 5.3 Security

* REQ‑NF3.1 All PII encrypted at rest (AES‑256‑GCM).
* REQ‑NF3.2 Annual penetration test; remediation within 30 days.

## 5.4 Software Quality Attributes

* Availability ≥ 99.5 % during registration windows (Gemino et al., 2020).
* Maintainability: MTTR < 4 h.
* Usability: SUS ≥ 80 in beta.
* Process adherence: Each sprint must meet entry/exit criteria per RUP guidelines (Tsui et al., 2022, Ch. 4).

# 6  Other Requirements

Audit logs were retained for 5 years; GDPR exports are on demand—internationalization placeholder for future multi-language UI.

# 7  References

Gemino, A., Reich, B. H., & Serrador, P. (2020). Agile, traditional, and hybrid approaches to project success: Is hybrid a poor second choice? \*Project Management Journal, 51\*(2), 177–194. https://doi.org/10.1177/8756972820973082

Ginikanwa, N. F. (2024). A comparative analysis of agile and traditional project management methodologies and their impact on information‑system development projects: A Nigerian banking system case study. \*International Journal of Science and Research, 13\*(8), 1682–1690.

Tsui, F., Karam, O., & Bernal, B. (2022). \*Essentials of software engineering\* (5th ed.). Jones & Bartlett Learning.