# CHAPTER 5: System Design

## 5.1. Architectural Design

### 5.1.1. Current Software Architecture

The UNIDOCS system follows a modern, layered architecture pattern that separates concerns and promotes maintainability. The current architecture is built using the **Spring Boot framework** for the backend, **Angular** for the web frontend, and **Flutter** for the mobile application, all communicating through RESTful APIs.

The system employs a **multi-tier architecture** consisting of:

1. **Presentation Layer**: Angular web application and Flutter mobile app
2. **API Layer**: REST controllers handling HTTP requests
3. **Business Logic Layer**: Service classes implementing business rules
4. **Data Access Layer**: Repository classes for database operations
5. **Data Layer**: PostgreSQL database and external services

### 5.1.2. Proposed Software Architecture

The proposed architecture enhances the current system with improved scalability, security, and performance. The architecture follows the **Microservices pattern** with clear separation of concerns and loose coupling between components.

#### Subsystem Decomposition

The UNIDOCS system is decomposed into the following subsystems:

1. **User Management Subsystem**
   * Handles user authentication, authorization, and profile management
   * Provides user registration, login, and password reset functionality
   * Manages role-based access control (Student vs Admin)
2. **Document Management Subsystem**
   * Manages letter request processing and approval workflows
   * Handles CV generation with AI integration
   * Provides document storage and retrieval capabilities
3. **Communication Subsystem**
   * Manages announcement creation and distribution
   * Handles email notifications and system alerts
   * Provides real-time communication features
4. **Analytics Subsystem**
   * Collects and processes system usage data
   * Generates reports and analytics for administrators
   * Provides insights into system performance and user behavior
5. **AI Integration Subsystem**
   * Integrates with PawaAI services for CV generation
   * Provides intelligent document processing capabilities
   * Handles AI-powered chat support

## 5.2. Database Design

### 5.2.1. Relational Model

The database design follows the **Third Normal Form (3NF)** to eliminate data redundancy and ensure data integrity. The relational model consists of five main entities with well-defined relationships:

**Primary Entities:**

* **USER**: Stores user information and authentication data
* **LETTER\_REQUEST**: Manages academic letter applications
* **CV\_REQUEST**: Handles CV generation requests
* **ANNOUNCEMENT**: Stores system announcements
* **NOTIFICATION**: Tracks email notifications sent to users

**Relationships:**

* USER (1) → LETTER\_REQUEST (M): One user can submit many letter requests
* USER (1) → CV\_REQUEST (M): One user can request many CVs
* USER (1) → ANNOUNCEMENT (M): One admin can create many announcements
* USER (1) → NOTIFICATION (M): One user can receive many notifications

### 5.2.2. Data Description

**USER Table:**

* Primary entity storing all user information
* Supports both student and admin roles
* Includes authentication and profile management fields
* Maintains password reset functionality

**LETTER\_REQUEST Table:**

* Stores comprehensive letter application details
* Includes personal information, academic details, and request specifics
* Tracks request status through approval workflow
* Links to user who submitted the request

**CV\_REQUEST Table:**

* Contains CV generation request information
* Stores educational background, experience, and skills
* Includes template selection and customization options
* Links to user who made the request

**ANNOUNCEMENT Table:**

* Manages system-wide announcements
* Includes title, content, and status information
* Tracks creation date and creator information
* Links to admin who created the announcement

**NOTIFICATION Table:**

* Records all email notifications sent by the system
* Tracks notification type, status, and delivery information
* Provides audit trail for communication history
* Links to user who received the notification

### 5.2.3. Data Dictionaries

| Table Name | Field Name | Data Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| USER | id | BIGINT | PRIMARY KEY | Unique user identifier |
| USER | email | VARCHAR(255) | UNIQUE, NOT NULL | User's email address |
| USER | first\_name | VARCHAR(100) | NOT NULL | User's first name |
| USER | last\_name | VARCHAR(100) | NOT NULL | User's last name |
| USER | role | VARCHAR(20) | NOT NULL | User role (student/admin) |
| USER | password | VARCHAR(255) | NOT NULL | Encrypted password |
| USER | status | VARCHAR(20) | NOT NULL | Account status |
| LETTER\_REQUEST | id | UUID | PRIMARY KEY | Unique request identifier |
| LETTER\_REQUEST | full\_name | VARCHAR(255) | NOT NULL | Student's full name |
| LETTER\_REQUEST | email | VARCHAR(255) | NOT NULL | Student's email |
| LETTER\_REQUEST | letter\_type | VARCHAR(50) | NOT NULL | Type of letter requested |
| LETTER\_REQUEST | status | VARCHAR(20) | NOT NULL | Request status |
| CV\_REQUEST | id | UUID | PRIMARY KEY | Unique CV request identifier |
| CV\_REQUEST | cv\_template | VARCHAR(50) | NOT NULL | Selected CV template |
| CV\_REQUEST | education | TEXT | NOT NULL | Educational background |
| CV\_REQUEST | experience | TEXT | NOT NULL | Work experience |
| ANNOUNCEMENT | id | BIGINT | PRIMARY KEY | Unique announcement identifier |
| ANNOUNCEMENT | title | VARCHAR(255) | NOT NULL | Announcement title |
| ANNOUNCEMENT | content | TEXT | NOT NULL | Announcement content |
| NOTIFICATION | id | BIGINT | PRIMARY KEY | Unique notification identifier |
| NOTIFICATION | type | VARCHAR(50) | NOT NULL | Notification type |
| NOTIFICATION | status | VARCHAR(20) | NOT NULL | Delivery status |

## 5.3. User Interface Design

### 5.3.1. Forms and Reports

The UNIDOCS system provides a comprehensive set of forms and reports designed for optimal user experience:

**Student Forms:**

* **Registration Form**: Collects student information during account creation
* **Letter Request Form**: Comprehensive form for submitting academic letter applications
* **CV Generation Form**: Multi-step form for CV creation with template selection
* **Profile Update Form**: Allows students to modify their personal information

**Admin Forms:**

* **Request Management Form**: Interface for reviewing and processing letter requests
* **Announcement Creation Form**: Rich text editor for creating system announcements
* **User Management Form**: Tools for managing student accounts and permissions
* **Analytics Dashboard**: Interactive reports and statistics display

**Reports:**

* **Request Status Report**: Shows current status of all letter requests
* **User Activity Report**: Tracks system usage and user engagement
* **Document Generation Report**: Statistics on CV and letter generation
* **Notification History Report**: Audit trail of all system communications

### 5.3.2. Interface Design Sample

[image]

The user interface follows modern design principles with a clean, intuitive layout. The design emphasizes:

* **Responsive Design**: Adapts to different screen sizes and devices
* **Accessibility**: WCAG 2.1 AA compliant with proper contrast and navigation
* **User-Friendly Navigation**: Clear menu structure and breadcrumb navigation
* **Consistent Styling**: Unified color scheme and typography throughout
* **Interactive Elements**: Hover effects, loading states, and feedback mechanisms

## 5.4. Access Control and Security

### 5.4.1. Role-Based Access Control (RBAC)

The UNIDOCS system implements a comprehensive role-based access control system with two primary roles:

**Student Role:**

* **Authentication**: Can register, login, and reset password
* **Profile Management**: Can view and update personal information
* **Letter Requests**: Can submit, view, and track letter applications
* **CV Generation**: Can create and download CV documents
* **Announcements**: Can view system announcements
* **Analytics**: Can view personal activity statistics

**Admin Role:**

* **User Management**: Can view, edit, and manage student accounts
* **Request Processing**: Can review, approve, or reject letter requests
* **Announcement Management**: Can create, edit, and delete announcements
* **System Analytics**: Can access comprehensive system reports
* **Document Management**: Can manage CV templates and generation settings
* **Notification Management**: Can view and manage system notifications

### 5.4.2. Security Measures

**Authentication Security:**

* **JWT Tokens**: Secure token-based authentication
* **Password Encryption**: BCrypt hashing for password storage
* **Session Management**: Secure session handling with automatic timeout

**Data Security:**

* **Data Encryption**: All sensitive data encrypted in transit and at rest
* **SQL Injection Prevention**: Parameterized queries and input validation

**Access Control:**

* **API Security**: REST API endpoints protected with authentication
* **Method-Level Security**: Fine-grained access control on service methods
* **Audit Logging**: Comprehensive logging of all system activities
* **Rate Limiting**: Protection against brute force attacks

### 5.4.3. Security Implementation

The security implementation follows industry best practices:

1. **Input Validation**: All user inputs are validated and sanitized
2. **Output Encoding**: All outputs are properly encoded to prevent XSS
3. **Error Handling**: Secure error messages that don't expose system details
4. **Logging and Monitoring**: Comprehensive audit trails for security events
5. **Regular Updates**: Security patches and updates applied regularly

This comprehensive security framework ensures that the UNIDOCS system maintains data integrity, protects user privacy, and provides secure access to all system functionalities while maintaining ease of use for legitimate users.

# CHAPTER 6: System Implementation and Testing

## 6.1. Technologies

The UNIDOCS system is implemented using a modern technology stack that ensures scalability, maintainability, and performance:

### 6.1.1. Backend Technologies

* **Java 17**: Core programming language for backend development
* **Spring Boot 3.3.1**: Framework for building production-ready applications
* **Spring Security**: Authentication and authorization framework
* **Spring Data JPA**: Data access layer with Hibernate ORM
* **PostgreSQL**: Primary relational database
* **JWT (JSON Web Tokens)**: Stateless authentication mechanism
* **Maven**: Build automation and dependency management

### 6.1.2. Frontend Technologies

* **Angular 16**: Frontend framework for web application
* **TypeScript**: Typed JavaScript for better development experience
* **HTML5/CSS3**: Markup and styling languages
* **Bootstrap**: CSS framework for responsive design
* **RxJS**: Reactive programming library for asynchronous operations

### 6.1.3. Mobile Technologies

* **Flutter 3.x**: Cross-platform mobile development framework
* **Dart**: Programming language for Flutter applications
* **Material Design**: UI/UX design system for mobile apps

### 6.1.4. External Services

* **PawaAI API**: AI-powered document generation services
* **SMTP Server**: Email service for notifications
* **Docker**: Containerization for deployment
* **Git**: Version control system

## 6.2. Database Implementation

### 6.2.1. Internal Schema of Database (Database Schema)

The database schema is implemented using PostgreSQL with the following structure:

-- Users table

CREATE TABLE users (

id BIGSERIAL PRIMARY KEY,

email VARCHAR(255) UNIQUE NOT NULL,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

password VARCHAR(255) NOT NULL,

role VARCHAR(20) NOT NULL DEFAULT 'student',

status VARCHAR(20) NOT NULL DEFAULT 'active',

reset\_token VARCHAR(255),

reset\_token\_expiry TIMESTAMP,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Letter requests table

CREATE TABLE letter\_requests (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

full\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

registration\_number VARCHAR(50) NOT NULL,

phone\_number VARCHAR(20) NOT NULL,

program\_of\_study VARCHAR(100) NOT NULL,

year\_of\_study INTEGER NOT NULL,

letter\_type VARCHAR(50) NOT NULL,

reason TEXT,

status VARCHAR(20) NOT NULL DEFAULT 'PENDING',

admin\_comment TEXT,

request\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id BIGINT REFERENCES users(id)

);

-- CV requests table

CREATE TABLE cv\_requests (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

full\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone\_number VARCHAR(20) NOT NULL,

address TEXT NOT NULL,

education TEXT NOT NULL,

experience TEXT NOT NULL,

skills TEXT NOT NULL,

cv\_template VARCHAR(50) NOT NULL,

about TEXT,

program\_of\_study VARCHAR(100),

submission\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id BIGINT REFERENCES users(id)

);

-- Announcements table

CREATE TABLE announcements (

id BIGSERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'active',

created\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id BIGINT REFERENCES users(id)

);

-- Notifications table

CREATE TABLE notifications (

id BIGSERIAL PRIMARY KEY,

email VARCHAR(255) NOT NULL,

subject VARCHAR(255) NOT NULL,

message TEXT NOT NULL,

type VARCHAR(50) NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'SENT',

sent\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id BIGINT REFERENCES users(id)

);

### 6.2.2. Data Dictionary

| Table | Column | Data Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| users | id | BIGSERIAL | PRIMARY KEY | Auto-incrementing user ID |
| users | email | VARCHAR(255) | UNIQUE, NOT NULL | User's email address |
| users | first\_name | VARCHAR(100) | NOT NULL | User's first name |
| users | last\_name | VARCHAR(100) | NOT NULL | User's last name |
| users | password | VARCHAR(255) | NOT NULL | Encrypted password hash |
| users | role | VARCHAR(20) | NOT NULL, DEFAULT 'student' | User role (student/admin) |
| users | status | VARCHAR(20) | NOT NULL, DEFAULT 'active' | Account status |
| letter\_requests | id | UUID | PRIMARY KEY | Unique request identifier |
| letter\_requests | full\_name | VARCHAR(255) | NOT NULL | Student's full name |
| letter\_requests | email | VARCHAR(255) | NOT NULL | Student's email |
| letter\_requests | letter\_type | VARCHAR(50) | NOT NULL | Type of letter requested |
| letter\_requests | status | VARCHAR(20) | NOT NULL, DEFAULT 'PENDING' | Request status |
| cv\_requests | id | UUID | PRIMARY KEY | Unique CV request identifier |
| cv\_requests | cv\_template | VARCHAR(50) | NOT NULL | Selected CV template |
| cv\_requests | education | TEXT | NOT NULL | Educational background |
| cv\_requests | experience | TEXT | NOT NULL | Work experience |
| announcements | id | BIGSERIAL | PRIMARY KEY | Auto-incrementing announcement ID |
| announcements | title | VARCHAR(255) | NOT NULL | Announcement title |
| announcements | content | TEXT | NOT NULL | Announcement content |
| notifications | id | BIGSERIAL | PRIMARY KEY | Auto-incrementing notification ID |
| notifications | type | VARCHAR(50) | NOT NULL | Notification type |
| notifications | status | VARCHAR(20) | NOT NULL, DEFAULT 'SENT' | Delivery status |

## 6.3. Testing

### 6.3.1. System Testing Methodology

The UNIDOCS system underwent comprehensive testing using multiple testing approaches:

**Unit Testing**: Individual components tested in isolation  
**Integration Testing**: Component interactions tested  
**System Testing**: End-to-end functionality testing  
**User Acceptance Testing**: Real-world scenario testing

### 6.3.2. Testing Results and Answers

#### Does each functionality of the system produce an expected output?

**YES** - All core functionalities produce expected outputs:

* **User Registration**: Successfully creates user accounts and sends confirmation emails
* **User Login**: Correctly authenticates users and redirects to appropriate dashboards
* **Letter Request Submission**: Properly stores requests and sends confirmation notifications
* **CV Generation**: Successfully generates CV documents using AI integration
* **Admin Request Management**: Correctly processes and updates request statuses
* **Announcement System**: Properly creates and displays announcements to students

#### Does the system respond with invalid data entry?

**YES** - The system implements comprehensive input validation:

* **Email Validation**: Rejects invalid email formats
* **Required Field Validation**: Prevents submission with missing mandatory fields
* **Data Type Validation**: Ensures correct data types for all inputs
* **Length Validation**: Enforces character limits on text fields
* **Format Validation**: Validates phone numbers, registration numbers, and dates

**Example Test Cases:**

* Invalid email format → System displays error message
* Missing required fields → Form submission blocked
* Invalid phone number → Validation error shown
* Empty content → Submission prevented

#### Does the system respond with incorrect login credentials?

**YES** - Robust authentication error handling:

* **Invalid Email**: System displays "User not found" message
* **Incorrect Password**: System shows "Invalid credentials" error
* **Inactive Account**: System prevents login and shows account status
* **Multiple Failed Attempts**: System implements rate limiting
* **Expired Tokens**: System redirects to login page

**Security Test Results:**

* Failed login attempts properly logged
* Account lockout after multiple failed attempts
* Secure error messages that don't reveal system information

#### Does your system restrict users from performing non-privileged functionalities?

**YES** - Comprehensive role-based access control:

**Student Restrictions:**

* Cannot access admin dashboard
* Cannot approve/reject requests
* Cannot create announcements
* Cannot view other users' data
* Cannot access system analytics

**Admin Privileges:**

* Full access to request management
* Ability to create and manage announcements
* Access to user management features
* System analytics and reporting access
* Document template management

**Security Test Results:**

* Unauthorized access attempts properly blocked
* Role-based UI elements correctly hidden/shown
* API endpoints properly secured with authentication

#### Did you implement the system using the proposed techniques and development approaches?

**YES** - Successfully implemented using proposed Object-Oriented and Bottom-Up approaches:

**Object-Oriented Implementation:**

* Clear class hierarchies and relationships
* Proper encapsulation of data and methods
* Inheritance used for user roles
* Polymorphism in service implementations

**Bottom-Up Development:**

* Individual components developed and tested first
* Progressive integration of components
* Parallel development of frontend and backend
* Incremental testing and validation

**Technology Stack Compliance:**

* Spring Boot backend as proposed
* Angular web frontend as planned
* Flutter mobile app as designed
* PostgreSQL database as specified
* AI integration with PawaAI as intended

## 6.4. User Interfaces

[image] - Student Login Interface  
[image] - Student Dashboard  
[image] - Letter Request Form  
[image] - CV Generation Interface  
[image] - Admin Dashboard  
[image] - Request Management Interface  
[image] - Announcement Creation Form  
[image] - Mobile App Login Screen  
[image] - Mobile Dashboard

## 6.5. Strengths and Limitations of the System

### 6.5.1. What is Covered from Requirements

**Successfully Implemented Features:**

1. **User Management System**
   * Complete user registration and authentication
   * Role-based access control (Student/Admin)
   * Password reset functionality
   * Profile management capabilities
2. **Letter Request Management**
   * Comprehensive letter application submission
   * Multiple letter types support
   * Request tracking and status updates
   * Admin approval/rejection workflow
3. **CV Generation System**
   * AI-powered CV generation with PawaAI
   * Multiple template options
   * Customizable content fields
   * Document download functionality
4. **Announcement System**
   * Admin announcement creation
   * Student announcement viewing
   * Rich text content support
   * Status management
5. **Communication System**
   * Email notifications for all major events
   * Notification tracking and history
   * Automated status updates
6. **Security Features**
   * JWT-based authentication
   * Input validation and sanitization
   * Role-based access control
   * Audit logging
7. **Multi-Platform Support**
   * Web application (Angular)
   * Mobile application (Flutter)
   * Responsive design

### 6.5.2. What is Not Covered

**Limitations and Future Enhancements:**

1. **Advanced Features**
   * Real-time chat support (planned for future)
   * Bulk operations for admins
   * Document versioning system
2. **Integration Limitations**
   * Limited third-party integrations
   * No payment gateway integration
   * No SMS notifications (email only)
3. **Scalability Considerations**
   * No load balancing implementation
   * Limited caching mechanisms
   * No microservices architecture
   * No advanced monitoring tools
4. **User Experience Enhancements**
   * No dark mode support
   * Limited customization options
   * No advanced search functionality
   * No export capabilities for reports
5. **Administrative Features**
   * No advanced user management tools
   * Limited backup and recovery options
   * No system configuration interface
   * No advanced audit trail features

**Future Development Roadmap:**

* Implementation of real-time notifications
* Advanced analytics dashboard
* Enhanced mobile app features
* Integration with additional AI services
* Advanced security features
* Performance optimization
* Enhanced user experience features

The system successfully meets all core requirements while providing a solid foundation for future enhancements and scalability improvements.

**CHAPTER 7: Conclusion, Recommendations and Challenges**

## ****7.1 Conclusion****

The development of the UNIDOCS system represents a significant step toward digitizing and streamlining the process of official document generation at the State University of Zanzibar. By leveraging modern technologies such as Spring Boot, Angular, and PostgreSQL, the system ensures secure, efficient, and user-friendly access to essential academic documents. This project has successfully addressed the manual inefficiencies previously observed in the university's document processing system, including delays, data inconsistencies, and lack of transparency. The implementation of features such as automated letter generation, CV building, request tracking, and real-time notifications reflects a user-centered design that aligns with the institution's goals for digital transformation.

## ****7.2 Recommendations****

To further enhance and sustain the impact of the UNIDOCS system, the following recommendations are proposed:

1. **Continuous User Training**: Regular workshops and training should be conducted for students and administrative staff to ensure effective use of the system.
2. **Integration with SUZA SIS and Email System**: Incorporating student information system (SIS) data and email notification integration will improve automation and reduce manual data entry.
3. **Data Backup and Recovery**: Implement a robust data backup and disaster recovery mechanism to prevent data loss and ensure system reliability.
4. **Security Enhancements**: Introduce multi-factor authentication (MFA) and routine security audits to protect user data and maintain privacy.
5. **Scalability Planning**: As student demand grows, the system should be tested and updated to handle higher loads without compromising performance.

## ****7.3 Challenges****

During the system development and deployment, several challenges were encountered, including:

* **Requirement Gathering**: Accurately capturing all user needs and expectations took more time than anticipated due to varied stakeholder inputs.
* **Technical Constraints**: Ensuring compatibility between Angular and Spring Boot, as well as maintaining performance, required careful system architecture planning.
* **Time Constraints**: The project had to be delivered within academic deadlines, limiting the opportunity for extended testing and optimization.
* **User Resistance**: Some users were initially resistant to adopt the digital system due to familiarity with traditional paper-based methods.

Despite these challenges, the system was successfully implemented with core functionalities and received positive feedback during pilot testing.