# Software Design Document (SDD) for [System Name]  
  
## Table of Contents  
  
1. [Introduction](#introduction)  
2. [System Overview](#system-overview)  
3. [Functional Requirements](#functional-requirements)  
4. [Non-Functional Requirements](#non-functional-requirements)  
5. [Technical Requirements](#technical-requirements)  
6. [API Design](#api-design)  
7. [Technology Stack](#technology-stack)  
8. [Wireframe Designs](#wireframe-designs)  
9. [Interface Validation Rules](#interface-validation-rules)  
10. [Security & Compliance](#security--compliance)  
11. [Data Management Strategy](#data-management-strategy)  
12. [Use Case Diagrams](#use-case-diagrams)  
13. [External System Integration Flow](#external-system-integration-flow)  
14. [Accessibility & Usability Compliance](#accessibility--usability-compliance)  
15. [Traceability Matrix](#traceability-matrix)  
16. [Appendices](#appendices)  
17. [Missing Requirements](#missing-requirements)  
  
## 1. Introduction  
  
### 1.1. Purpose  
  
This Software Design Document (SDD) defines the design for the [System Name] system. It elaborates on the requirements outlined in the Software Requirements Specification (SRS) document and provides a detailed blueprint for the system's development. This document serves as a guide for developers, testers, and other stakeholders involved in the project.  
As outlined in the SRS: To define the requirements for the [System Name] system.  
  
### 1.2. Scope  
  
This document covers all aspects of the [System Name] system's design, including the system architecture, database schema, user interface, API specifications, security considerations, deployment strategy, accessibility, and usability considerations.  
As outlined in the SRS: This document covers all functional and non-functional requirements for the [System Name] system, including user interfaces, API integrations, security considerations, and data management strategies.  
  
### 1.3. Assumptions  
  
\* The development team possesses the necessary skills and expertise in the chosen technology stack.  
\* Stakeholders will actively participate in the design review process and provide timely feedback.  
\* Third-party APIs and services will be reliable and available as required.  
As outlined in the SRS:  
 \* The development team has sufficient expertise in the chosen technology stack.  
 \* Stakeholders will provide timely feedback and approvals.  
 \* Third-party APIs will remain stable and available.  
  
### 1.4. Constraints  
  
\* \*\*Budget:\*\* The project must be completed within a budget of [Amount].  
\* \*\*Timeline:\*\* The system must be deployed by [Date].  
\* \*\*Technology:\*\* The system must be compatible with existing infrastructure, including [Specify].  
As outlined in the SRS:  
 \* Budget limitations: The project must be completed within a budget of [Amount].  
 \* Timeline constraints: The system must be deployed by [Date].  
 \* Technology constraints: The system must be compatible with existing infrastructure, including [Specify].  
  
### 1.5. Stakeholders  
  
| Name | Role | Responsibilities | Contact Information |  
| ------------------- | --------------------- | --------------------------------------------------------- | ------------------------- |  
| [Stakeholder Name] | [Stakeholder Role] | [Stakeholder Responsibilities] | [Stakeholder Contact Information] |  
  
As outlined in the SRS:  
 \* Stakeholders:  
 {  
 "Name": "[Stakeholder Name]",  
 "Role": "[Stakeholder Role]",  
 "Responsibilities": "[Stakeholder Responsibilities]",  
 "Contact\_Information": "[Stakeholder Contact Information]"  
 }  
  
### 1.6. Glossary  
  
| Term | Definition |  
| --------- | -------------------------------------------------------------- |  
| [Term] | [Definition] |  
  
As outlined in the SRS:  
 \* Glossary:  
 {  
 "Term": "Definition"  
 }  
  
## 2. System Overview  
  
### 2.1. High-Level Architecture  
  
The [System Name] system adopts a three-tier architecture, comprising a presentation tier (user interface), an application tier (business logic), and a data tier (database). Users interact with the system through a web-based interface. The application tier handles user requests, processes data, and interacts with the database. The data tier stores and retrieves persistent data. Communication between tiers occurs via APIs.  
  
```mermaidstart  
graph LR  
 A[User] --> B(Web Server);  
 B --> C{Application Server};  
 C --> D[(Database)];  
 C --> E[External API];  
 style A fill:#f9f,stroke:#333,stroke-width:2px  
 style B fill:#ccf,stroke:#333,stroke-width:2px  
 style C fill:#ccf,stroke:#333,stroke-width:2px  
 style D fill:#ccf,stroke:#333,stroke-width:2px  
 style E fill:#ccf,stroke:#333,stroke-width:2px  
```mermaidend  
  
As outlined in the SRS: A high-level diagram and description of the system's components and their interactions.  
  
### 2.2. Technology Stack  
  
\* \*\*Programming Languages:\*\* Python 3.9  
\* \*\*Frameworks:\*\* Django 3.2, React 17  
\* \*\*Databases:\*\* PostgreSQL 13  
\* \*\*Cloud Platform:\*\* AWS (Amazon Web Services)  
  
As outlined in the SRS:  
 \* Programming Languages: [List]  
 \* Frameworks: [List]  
 \* Databases: [List]  
 \* Cloud Platform: [Platform Name]  
  
### 2.3. Deployment Architecture  
  
The system will be deployed on a cloud platform (AWS). The deployment will utilize a load balancer to distribute traffic across multiple web servers. Application servers will handle the business logic and data processing. The database will be deployed in a clustered configuration for high availability and scalability. Docker containers will be used for deployment and scaling.  
  
```mermaidstart  
graph LR  
 A[Load Balancer (AWS ALB)] --> B(Web Server 1 (EC2));  
 A --> C(Web Server 2 (EC2));  
 B --> D{Application Server 1 (ECS)};  
 C --> E{Application Server 2 (ECS)};  
 D --> F[(Database (RDS PostgreSQL))];  
 E --> F;  
 style A fill:#f9f,stroke:#333,stroke-width:2px  
 style B fill:#ccf,stroke:#333,stroke-width:2px  
 style C fill:#ccf,stroke:#333,stroke-width:2px  
 style D fill:#ccf,stroke:#333,stroke-width:2px  
 style E fill:#ccf,stroke:#333,stroke-width:2px  
 style F fill:#ccf,stroke:#333,stroke-width:2px  
```mermaidend  
As outlined in the SRS: Description of how the system will be deployed, including servers, network configurations, and load balancing strategies.  
  
## 3. Functional Requirements  
  
To provide concrete examples, let's assume the following functional requirement.  
  
```json  
{  
 "Requirement\_ID": "FR001",  
 "Description": "The system shall allow users to log in with a valid username and password.",  
 "Priority": "High",  
 "Input": "Username, Password",  
 "Output": "Successful login and redirection to the dashboard, or display of an error message.",  
 "Preconditions": "The user must have a registered account.",  
 "Postconditions": "The user session is established, and the user is authenticated.",  
 "Acceptance\_Criteria": "Users can successfully log in with valid credentials and are redirected to the dashboard. Invalid credentials result in an appropriate error message."  
}

### 3.1 FR001 - User Login

* **Description:** The system shall allow users to log in with a valid username and password.
* **Priority:** High
* **Input:** Username, Password
* **Output:** Successful login and redirection to the dashboard, or display of an error message.
* **Preconditions:** The user must have a registered account.
* **Postconditions:** The user session is established, and the user is authenticated.
* **Acceptance Criteria:** Users can successfully log in with valid credentials and are redirected to the dashboard. Invalid credentials result in an appropriate error message.

### 3.2 FR002 - User Registration

* **Description:** The system shall allow new users to register with a valid email address, username and password.
* **Priority:** High
* **Input:** Email, Username, Password, Confirm Password
* **Output:** Successful registration and redirection to the login page, or display of an error message.
* **Preconditions:** None.
* **Postconditions:** The user account is created and stored in the database.
* **Acceptance Criteria:** Users can successfully register with valid credentials and are redirected to the login page. Invalid credentials result in an appropriate error message. The password and confirm password fields must match.

## 4. Non-Functional Requirements

### 4.1. Performance

| Requirement ID | Description | Metric | Measurement |
| --- | --- | --- | --- |
| NF001 | The system shall respond to user requests within [Time] seconds. | Response Time | [Time Unit] |

As outlined in the SRS: \* Requirement\_ID”: “NF001”, \* “Description”: “The system shall respond to user requests within [Time] seconds.”, \* “Metric”: “[Response Time]”, \* “Measurement”: “[Time Unit]”

### 4.2. Scalability

| Requirement ID | Description | Metric | Measurement |
| --- | --- | --- | --- |
| NF002 | The system shall be able to handle [Number] concurrent users without degradation. | Concurrent Users | [Number] |

As outlined in the SRS: \* Requirement\_ID”: “NF002”, \* “Description”: “The system shall be able to handle [Number] concurrent users without performance degradation.”, \* “Metric”: “[Concurrent Users]”, \* “Measurement”: “[Number]”

### 4.3. Usability

| Requirement ID | Description |
| --- | --- |
| NF003 | The system shall have a user-friendly interface that is easy to navigate. |

As outlined in the SRS: \* Requirement\_ID”: “NF003”, \* “Description”: “The system shall have a user-friendly interface that is easy to navigate.”

### 4.4. Reliability

| Requirement ID | Description |
| --- | --- |
| NF004 | The system shall have an uptime of 99.9%. |

As outlined in the SRS: \* Requirement\_ID”: “NF004”, \* “Description”: “The system shall have an uptime of 99.9%.”

### 4.5. Maintainability

| Requirement ID | Description |
| --- | --- |
| NF005 | The system shall be modular and well-documented to facilitate future maintenance and updates. |

As outlined in the SRS: \* Requirement\_ID”: “NF005”, \* “Description”: “The system shall be modular and well-documented to facilitate future maintenance and updates.”

### 4.6. Portability

| Requirement ID | Description |
| --- | --- |
| NF006 | The system shall be portable across different environments (e.g., cloud, on-premise). |

As outlined in the SRS: \* Requirement\_ID”: “NF006”, \* “Description”: “The system shall be portable across different environments (e.g., cloud, on-premise).”

## 5. Technical Requirements

* **Coding Standards:** Code must adhere to established style guides (e.g., PEP 8 for Python, Google Java Style for Java).
* **Testing Procedures:** Unit tests, integration tests, and system tests will be conducted. A test-driven development (TDD) approach is recommended.
* **Integration Requirements:** The system must integrate with [Specify external system] via [Specify API or protocol].
* **Accessibility:** The system must adhere to WCAG 2.1 Level AA guidelines.
* **Security:** All data transmitted over the network must be encrypted using TLS 1.3 or higher.
* **Data Encryption:** All data at rest must be encrypted using AES-256 encryption. Key management will follow industry best practices, using a Hardware Security Module (HSM).

## 6. API Design

### 6.1. API Endpoints

| Endpoint | Method | Description | Request Parameters | Response Format | Authentication |
| --- | --- | --- | --- | --- | --- |
| /api/login | POST | Authenticates a user | username, password | JSON | JWT |
| /api/register | POST | Registers a new user | email, username, password | JSON | None |
| /api/users/{userID} | GET | Retrieves user information | userID | JSON | JWT |

### 6.2. External Integration Points

| System | Purpose | Data Exchange Format | Authentication Method |
| --- | --- | --- | --- |
| [External System Name] | [Purpose of Integration] | [Data Format] | [Authentication Method] |

## 7. Technology Stack

* **Programming Languages:** Python 3.9, JavaScript (ES6+)
* **Frameworks:** Django 3.2 (Backend), React 17 (Frontend), Django REST Framework
* **Databases:** PostgreSQL 13
* **Cloud Platform:** AWS (Amazon Web Services)
  + EC2
  + ECS
  + RDS
  + ALB
* **Libraries:**
  + axios (HTTP client for React)
  + bcrypt (password hashing)
  + jsonwebtoken (JWT authentication)
  + psycopg2 (PostgreSQL adapter for Python)

## 8. Wireframe Designs

### 8.1. Login Page

* **Description:** The login page allows users to enter their username and password to access the system.
* **Elements:**
  + Username Input Field
  + Password Input Field
  + Login Button
  + “Forgot Password” Link
  + “Create Account” Link
* **Navigation:**
  + Successful Login -> Dashboard
  + “Forgot Password” Link -> Password Reset Page
  + “Create Account” Link -> Registration Page
* **Accessibility Considerations:**
  + Use ARIA attributes to improve screen reader support.
  + Ensure sufficient color contrast between text and background.
  + Provide keyboard navigation support.
  + Use labels associated with input fields.

graph LR  
 A[Username Input] --> B(Password Input);  
 B --> C{Login Button};  
 C -- Success --> D[Dashboard];  
 C -- Failure --> E{Error Message};  
 C --> F[Forgot Password Link];  
 C --> G[Create Account Link];  
 F --> H[Password Reset Page];  
 G --> I[Registration Page];  
 style A fill:#f9f,stroke:#333,stroke-width:2px  
 style B fill:#ccf,stroke:#333,stroke-width:2px  
 style C fill:#ccf,stroke:#333,stroke-width:2px  
 style D fill:#ccf,stroke:#333,stroke-width:2px  
 style E fill:#ccf,stroke:#333,stroke-width:2px  
 style F fill:#ccf,stroke:#333,stroke-width:2px  
 style G fill:#ccf,stroke:#333,stroke-width:2px  
 style H fill:#ccf,stroke:#333,stroke-width:2px  
 style I fill:#ccf,stroke:#333,stroke-width:2px  
```mermaidend  
  
### 8.2. Registration Page  
  
\* \*\*Description:\*\* The registration page allows new users to create an account.  
\* \*\*Elements:\*\*  
 \* Email Input Field  
 \* Username Input Field  
 \* Password Input Field  
 \* Confirm Password Input Field  
 \* Register Button  
 \* "Already have an account" Link  
\* \*\*Navigation:\*\*  
 \* Successful Registration -> Login Page  
 \* "Already have an account" Link -> Login Page  
\* \*\*Accessibility Considerations:\*\*  
 \* Use ARIA attributes to improve screen reader support.  
 \* Ensure sufficient color contrast between text and background.  
 \* Provide keyboard navigation support.  
 \* Use field labels associated with the input fields.  
 \* Provide clear error messages.  
  
```mermaidstart  
graph LR  
 A[Email Input] --> B(Username Input);  
 B --> C(Password Input);  
 C --> D(Confirm Password Input);  
 D --> E{Register Button};  
 E -- Success --> F[Login Page];  
 E -- Failure --> G{Error Message};  
 E --> H[Already have an account Link];  
 H --> F;  
 style A fill:#f9f,stroke:#333,stroke-width:2px  
 style B fill:#ccf,stroke:#333,stroke-width:2px  
 style C fill:#ccf,stroke:#333,stroke-width:2px  
 style D fill:#ccf,stroke:#333,stroke-width:2px  
 style E fill:#ccf,stroke:#333,stroke-width:2px  
 style F fill:#ccf,stroke:#333,stroke-width:2px  
 style G fill:#ccf,stroke:#333,stroke-width:2px  
 style H fill:#ccf,stroke:#333,stroke-width:2px  
```mermaidend  
  
## 9. Interface Validation Rules  
  
### 9.1. Login Page Validation  
  
| Element | Validation Rule | Error Message |  
| -------------- | --------------- | --------------------------- |  
| Username Input | Required | "Please enter your username." |  
| Password Input | Required | "Please enter your password." |  
| Password Input | Minimum Length 8 characters | "Password must be at least 8 characters long." |  
  
### 9.2. Registration Page Validation  
  
| Element | Validation Rule | Error Message |  
| ------------------- | -------------------------------------------------------- | ---------------------------------------------------------------- |  
| Email Input | Required, Valid Email Format | "Please enter a valid email address." |  
| Username Input | Required, Minimum Length 3 characters | "Please enter a username with at least 3 characters." |  
| Password Input | Required, Minimum Length 8 characters, Strong Password | "Password must be at least 8 characters long and meet complexity requirements: at least one uppercase letter, one lowercase letter, one number, and one symbol." |  
| Confirm Password Input | Required, Must match Password Input | "Passwords do not match." |  
  
### 9.3. User Login Flow  
  
```mermaidstart  
stateDiagram  
 [\*] --> LoginPage  
 LoginPage : Enter Username and Password  
 LoginPage --> LoginPage : Invalid Credentials  
 LoginPage --> Dashboard : Valid Credentials  
 Dashboard : User is logged in  
```mermaidend  
  
### 9.4. User Registration Flow  
  
```mermaidstart  
stateDiagram  
 [\*] --> RegistrationPage  
 RegistrationPage : Enter Registration Details  
 RegistrationPage --> RegistrationPage : Invalid Input  
 RegistrationPage --> LoginPage : Successful Registration  
 LoginPage : Redirect to Login Page  
```mermaidend  
  
## 10. Security & Compliance  
  
### 10.1. Compliance Standards  
  
\* HIPAA  
\* GDPR  
\* OWASP Top 10  
\* WCAG 2.1 Level AA  
  
### 10.2. Security Best Practices  
  
\* Implement strong password policies. Passwords must be at least 8 characters long and include a mix of uppercase letters, lowercase letters, numbers, and symbols.  
\* Use encryption for sensitive data both in transit (TLS 1.3 or higher) and at rest (AES-256).  
\* Regularly update software and libraries.  
\* Use parameterized queries to prevent SQL injection.  
\* Implement CSRF protection using tokens.  
\* Implement output encoding to prevent cross-site scripting (XSS) attacks.  
\* Implement robust authentication and authorization mechanisms, such as multi-factor authentication (MFA).  
\* Conduct regular security audits and penetration testing.  
\* Implement logging and monitoring to detect and respond to security incidents.  
\* Implement rate limiting to protect against brute force attacks.  
\* Store password hashes using bcrypt with a high cost factor.  
\* Implement account lockout after multiple failed login attempts.  
\* Enforce MFA for privileged accounts.  
  
### 10.3. Threat Modeling  
  
| Threat | Likelihood | Impact | Mitigation | Compliance Standard |  
| ----------- | ---------- | ------ | ------------------------- | ------------------- |  
| SQL Injection | High | High | Use parameterized queries | OWASP |  
| XSS | High | High | Implement output encoding, Content Security Policy (CSP) | OWASP |  
| Brute Force Attack | Medium | High | Implement rate limiting, MFA, account lockout | OWASP, GDPR |  
| Data Breach involving PHI | Medium | High | Data encryption, access controls, audit logging | HIPAA, GDPR |  
| Data Breach involving PII | Medium | High | Data encryption, access controls, audit logging, data minimization | GDPR |  
| CSRF | Medium | High | Implement CSRF tokens | OWASP |  
| Insecure Direct Object Reference (IDOR) | Medium | High | Implement proper authorization checks | OWASP |  
  
### 10.4. Risk Assessment  
  
| Risk | Likelihood | Impact | Mitigation | Compliance Standard |  
| ----------- | ---------- | ------ | ------------------------- | ------------------- |  
| Unauthorized access to PHI | Medium | High | Implement RBAC, encryption, and audit logging, enforce MFA for privileged accounts, regularly review access controls | HIPAA |  
| Unauthorized access to PII | Medium | High | Implement RBAC, encryption, and audit logging, data minimization, regularly review access controls | GDPR |  
| Data loss due to system failure | Low | High | Implement data backup and recovery procedures, implement RAID for data storage | HIPAA, GDPR |  
| Data corruption | Low | High | Implement data validation and integrity checks, use checksums for data integrity | HIPAA, GDPR |  
| Phishing Attack | Medium | High | User Training, Spam Filters, MFA | GDPR |  
| Malware Infection | Medium | High | Antivirus Software, Regular Security Scans | GDPR, HIPAA |  
  
### 10.5. RBAC  
  
| Role | Permissions | Description |  
|---|---|---|  
| Administrator | Create, Read, Update, Delete users, roles, and system configuration, access all data | Full access to the system |  
| User | Read and Update own profile information, access own data | Limited access to the system |  
| Auditor | Read logs and audit trails, generate reports | Read-only access to audit data |  
  
### 10.6 Incident Response Plan  
  
In the event of a security incident, the following steps will be taken:  
1. \*\*Identification:\*\* Identify the scope and nature of the incident.  
2. \*\*Containment:\*\* Isolate affected systems to prevent further damage.  
3. \*\*Eradication:\*\* Remove the cause of the incident (e.g., malware, vulnerability).  
4. \*\*Recovery:\*\* Restore systems and data to normal operation.  
5. \*\*Lessons Learned:\*\* Document the incident and identify areas for improvement.  
6. \*\*Notification:\*\* Notify relevant stakeholders (e.g., affected users, regulatory bodies) as required by compliance standards (e.g., GDPR, HIPAA).  
  
### 10.7 Data Privacy Impact Assessment (DPIA)  
A DPIA will be conducted to identify and assess the privacy risks associated with the system. The DPIA will consider the following factors:  
\* The nature, scope, context and purposes of the processing  
\* The risks to the rights and freedoms of natural persons  
\* The measures envisaged to address the risks  
The DPIA will be reviewed and updated regularly.  
  
### 10.8 Password Complexity Requirements  
  
Passwords must meet the following complexity requirements:  
\* Minimum length: 12 characters  
\* At least one uppercase letter  
\* At least one lowercase letter  
\* At least one number  
\* At least one symbol  
\* Must not be a commonly used password  
\* Must not be based on personal information  
  
## 11. Data Management Strategy  
  
### 11.1. ER Schema  
  
[Describe the entity-relationship schema for the system's data. Include a diagram illustrating the schema.]  
  
```mermaidstart  
erDiagram  
 CUSTOMER ||--o{ ORDER : places  
 ORDER ||--|{ LINE-ITEM : contains  
 CUSTOMER {  
 string custId PK  
 string name  
 string address  
 }  
 ORDER {  
 string orderId PK  
 string orderDate  
 string custId FK  
 }  
 LINE-ITEM {  
 int lineItemId PK  
 int orderId FK  
 int productId FK  
 int quantity  
 }  
 PRODUCT {  
 string productId PK  
 string name  
 float price  
 }  
 LINE-ITEM ||--|| PRODUCT : has  
 USER {  
 string userID PK  
 string username  
 string password  
 string email  
 }  
 CUSTOMER ||--o{ USER : registered  
```mermaidend  
  
### 11.2. Data Retention Policy  
  
PII and PHI data will be retained only as long as necessary for the purposes for which it was collected, in accordance with GDPR and HIPAA requirements. User data will be anonymized or deleted after [Number] years of inactivity (e.g., 2 years). Audit logs will be retained for [Number] years (e.g. 5 years). Specific data retention periods will be defined for different data types, considering legal and regulatory requirements.  
  
### 11.3. Data Backup & Recovery  
  
Regular backups of the database will be performed and stored securely offsite. Backups will be encrypted and access-controlled. Backups will be tested regularly to ensure that they can be restored successfully in the event of a disaster. The Recovery Time Objective (RTO) will be [Time] (e.g., 4 hours) and the Recovery Point Objective (RPO) will be [Time] (e.g., 1 hour). Multiple backup copies will be maintained in geographically diverse locations.  
  
## 12. Use Case Diagrams  
  
[Include use case diagrams illustrating the interactions between users and the system.]  
  
```mermaidstart  
 @startuml  
left to right direction  
actor User  
rectangle System {  
 User -- (Login)  
 User -- (Register)  
 User -- (View Dashboard)  
 User -- (Edit Profile)  
}  
@enduml  
```mermaidend  
  
## 13. External System Integration Flow  
  
[Describe the flow of data and interactions between the system and any external systems.]  
  
```mermaidstart  
sequenceDiagram  
 participant User  
 participant System  
 participant ExternalAuth  
 User->>System: Login Request  
 System->>ExternalAuth: Authenticate User  
 ExternalAuth-->>System: Auth Confirmation  
 System->>User: Login Confirmation  
```mermaidend  
  
## 14. Accessibility & Usability Compliance  
  
\* \*\*WCAG 2.1 Level AA Compliance:\*\* The system will be designed and developed to meet WCAG 2.1 Level AA guidelines.  
\* \*\*Keyboard Navigation:\*\* All interactive elements will be accessible via keyboard navigation.  
\* \*\*Screen Reader Compatibility:\*\* The system will be compatible with popular screen readers (e.g., NVDA, JAWS).  
\* \*\*Color Contrast:\*\* Sufficient color contrast will be provided between text and background elements (minimum contrast ratio of 4.5:1 for normal text).  
\* \*\*Usability Testing:\*\* Usability testing will be conducted with a diverse group of users, including users with disabilities, to identify and address any usability issues.  
\* \*\*Assistive Technology Support:\*\* The system will be designed to work seamlessly with assistive technologies.  
\* \*\*Alternative Text:\*\* Appropriate alternative text will be provided for all images and non-text content.  
\* \*\*Form Labels:\*\* Form elements will have clear and descriptive labels.  
  
## 15. Traceability Matrix  
  
[A table that maps the requirements from the SRS document to the design elements in the SDD.]  
  
| Requirement ID (SRS) | Design Element (SDD) | Description | HIPAA | GDPR | OWASP | Test Case ID | Status |  
|---|---|---|---|---|---|---|---|  
| FR001 | 3.1 User Login | Functional requirement for user login | | | X | TC-FR001-01 | Implemented |  
| FR001 | 9.1 Login Page Validation | Interface Validation Rules for Login Page| | | X | TC-FR001-02 | Implemented |  
| FR002 | 3.2 User Registration | Functional requirement for user registration | | X | X | TC-FR002-01 | Implemented |  
| FR002 | 9.2 Registration Page Validation | Interface Validation Rules for Registration Page| | X | X | TC-FR002-02 | Implemented |  
| NF003 | 4.3 Usability | Non-functional requirement for usability | | | | | |  
| NF003 | 8.1 Login Page Accessibility Considerations | Accessibility considerations for login page | | | | | |  
| NF003 | 8.2 Registration Page Accessibility Considerations | Accessibility considerations for registration page | | | | | |  
| NF003 | 14. Accessibility & Usability Compliance | Details on achieving usability and accessibility compliance | | | | | |  
| SEC001 (New) | 10.2 Security Best Practices | Password Policy Implementation | X | X | X | TC-SEC001-01 | Implemented |  
| SEC002 (New) | 6.1 API Endpoints | Secure API Authentication and Authorization | X | X | X | TC-SEC002-01 | Implemented |  
| DM001 (New) | 11.2 Data Retention Policy | Defines data retention periods | X | X | | | |  
| DM002 (New) | 11.3 Data Backup & Recovery | Implements secure backup and recovery procedures | X | X | | | |  
| RBAC001 (New) | 10.5 RBAC | Defines roles and permissions | X | X | X | | |  
| WCAG001 (New) | 14. Accessibility & Usability Compliance | Implement WCAG guidelines | X | X | | | |  
  
## 16. Appendices  
  
\* \*\*Data Dictionary:\*\* Detailed descriptions of data elements and their formats.  
\* \*\*API Documentation:\*\* Comprehensive documentation of all APIs using OpenAPI/Swagger specification.  
\* \*\*Configuration Details:\*\* Detailed configuration information for all system components.  
\* \*\*Database Schema:\*\* Detailed database schema diagrams and descriptions.  
\* \*\*Deployment Scripts:\*\* Scripts for automating the deployment process.  
  
## 17. Missing Requirements  
  
\* Detailed API documentation  
\* Specific password complexity requirements.  
\* Detailed data retention policies.  
\* Data Loss Prevention (DLP) strategy  
\* Detailed information on third-party integrations and their security implications.  
\* Specific disaster recovery plan and procedures.  
\* Details on monitoring and alerting strategy.  
\* Details for session management and invalidation.  
\* Formal code review process and coding standards enforcement.  
\* Detailed description of data validation and sanitization techniques employed.  
\* Details on the frameworks and libraries that will be used.