****

**ST. PAUL’S UNIVERSITY**

**(LIMURU CAMPUS)**

**FACULITY OF COMPUTER SCIENCE AND COMMUNICATION STUDIES**

**BCS 3106: SOFTWARE ENGINEERING II**

**ASSIGNMENT: INITIAL DOCUMENTS FOR SEMESTER PROJECT**

**GROUP E:**

**BSCLMR110623**

**BSCLMR151323**

**BSCLMR150723**

**BSCLMR328421**

**BSCLMR101721**

**BSCLMR106823**

**GROUP GITHUB LINK:**https://github.com/philmanga/Group-E-SE2.git

**LECTURER(S): Cecilia Nanfuka**

**DATE OF SUBMISSION: 8th March, 2024**

**Software architecture and design for University Nomination Election System**

**High-Level Overview:**

* The application will be a web-based system for conducting university nomination elections.
* Through this application,students to nominate candidates, view candidate profiles and cast votes.
* Built using Flask( Python web framework).

High level choice**: Layered architecture**

This architecture assumes a proper way of doing things which includes specialized layers with specific functions

**Major components**

1. **Presentation layer**

Web interface developed with Flask.

Handles user interactions

Ensures business rules and security constrains are met.

**2.Data layer**

Uses database to store student informations, nominations and votes.

Securely access and update data.

**3.Security layer**

Manages user authentication,authorization and encrypts communication between them.

Implements strong security measures to secure sensitive data and prevent unauthorized access.

**4.Business logic layer**

Manages incoming user requests in relation to the data layer.

Performs nomination,voting functionalities and access control.

Ensures compliance with business rules and security constrains.

**Interactions**:

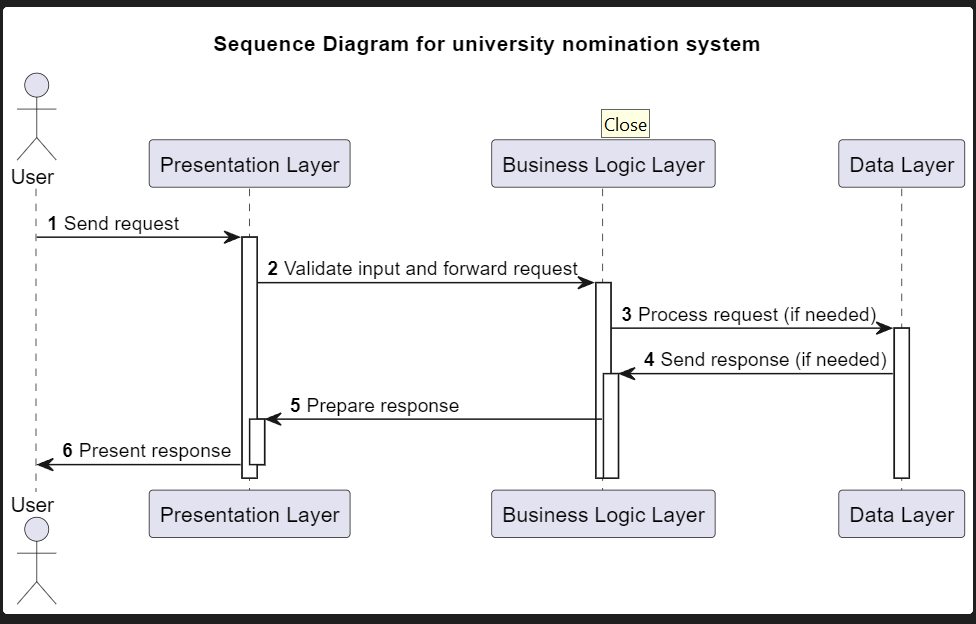
1.User interacts with the web interface(Presentation layer)

2.Presentation layer validates input and sends requests to the Business logic layer

3.Business logic layer processes the requests,interacts with the Data layer using the ORM,and enforces security rules.

4.Data layer receives or stores in the database

5.Result are sent back through the layers through layers to the Presentation layer for display.



**Data Architecture**

****Data Storage:****

****Database Type:**** Relational database as it is suitable due to its flexible schema and strong transaction support.

****Schema Design:****

* Normalized tables are used to minimize redundancy and improve data integrity.
* Foreign key constraints are defined to enforce relationships between entities.
* Each attribute should have appropriate data type.

**Entities**

1. **Voter**: This entity stores the information about the eligible voters, such as their ID, name, address, and voting history. The primary key is the voter ID.
2. **Candidate**: This entity stores the information about the candidates who are running for nomination, such as their ID, name, party, and platform. The primary key is the candidate ID.
3. **Nomination**: This entity stores the information about the nomination process, such as the type, date, location, and rules. The primary key is the nomination ID.
4. **Ballot**: This entity stores the information about the ballots used in the nomination, such as the ballot ID, nomination ID, and the list of candidates. The primary key is the ballot ID.
5. **Vote**: This entity stores the information about the votes cast by the voters, such as the vote ID, voter ID, ballot ID, and the candidate ID. The primary key is the vote ID.

These entities can be related by the following tables:

**Voter\_Candidate**: This table stores the relationship between the voters and the candidates they support. The primary key is the voter ID and candidate ID.

**Candidate\_Nomination**: This table stores the relationship between the candidates and nominations they participate in. Candidate ID and nomination ID combine to form this tables primary key.

**Nomination\_Ballot**: This table stores the relationship between the nominations and the ballots used in them. The primary key is the composite key of nomination ID and ballot ID.

**Ballot\_Vote**: This table stores the relationship between the ballots and the votes cast using them.Ballot ID and vote ID make up its primary key.

****Data Integrity****

1. Use of constraints (e.g.,foreign keys) to ensure data validity and consistency.
2. Implement data validation rules on both client-side and server-side.
3. Use triggers to maintain data integrity across tables.

**Security Architecture**

**Security threats**

1. ****Unauthorized Access:**** Malicious actors attempting to gain access to student data, nominations, or votes.
2. ****Data Tampering:**** Manipulation of nominations, votes, or student data to influence election outcomes.
3. ****Denial-of-Service (DoS) Attacks:**** Overwhelming the system with traffic to disrupt nominations or voting.
4. ****Man-in-the-Middle Attacks:**** Intercepting or modifying communication between users and the system.
5. ****Insider Threats:**** Authorized users abusing their access to compromise the system.

**Mitigation Strategies**

1. ****Authorization:**** Enforce Role-Based Access Control to restrict unauthorized access to data and functionalities.
2. ****Authentication:**** Implement strong password hashing and consider two-factor authentication for additional security.
3. ****Secure Communication:**** Use HTTPS to encrypt all communication between users and the server.
4. ****Encryption:**** Encrypt sensitive data (e.g., student IDs, votes) at rest and in transit using strong algorithms .
5. ****Input Validation:**** Sanitize and validate all user input to prevent SQL injection and other vulnerabilities.
6. ****Regular Security Audits:**** Conduct penetration testing and security assessments to identify and address vulnerabilities.
7. ****Logging and Monitoring:**** Log all user activities and system events for auditing and potential investigations.
8. ****Data Backups and Disaster Recovery:**** Regularly backing up of data and implementing disaster recovery plans to ensure system availability in case of failures.

**Deployment Architecture**

****Deployment Infrastructure****

****On-premises deployment:****

*Data control*: Offers complete control over data and infrastructure, which might be important for universities with strict data privacy requirements.It also requires managing your own hardware, software and security infrastructure.

*Customization*:Offers flexibility to customize the application and infrastructure to meet our specific needs.

**Deployment Environment**

1. ****Web Server:**** Use a production-grade web server like Apache, configured to serve your Flask application static files and handle reverse proxy connections if needed.
2. ****Database:**** Use of a robust database like MySQL to store user data, nominations, and other relevant information. Ensure the database is secure and adheres to data privacy regulations.
3. ****Additional Services:**** Implement services like caching for improved performance, email notifications, and security measures like firewalls and intrusion detection systems.

**Deployment Process**

1. **Version control**: Represents Git repository to track code changes and facilitate deployment.
2. ****Containerization:**** Use containerization technologies like Docker to package your application code and dependencies, ensuring consistent deployment across environments.
3. ****CI/CD Pipeline:**** Implement a continuous integration and continuous delivery (CI/CD) pipeline to automate building, testing, and deploying your application updates. This reduces manual errors and promotes rapid deployment.
4. ****Monitoring & Logging:**** Set up comprehensive monitoring and logging solutions to track application performance, user activity, and potential errors. This helps identify and address issues promptly.

**Scaling**

1. ****Horizontal scaling:**** Add more server instances behind a load balancer to distribute traffic and increase capacity during peak usage periods.
2. ****Vertical scaling:**** Upgrade server resources like CPU, RAM, and storage on existing instances to handle heavier workloads.

**High availability**:

1. Use a load balancer to distribute traffic across multiple server instances, ensuring continued service even if one instance fails.
2. Implement database replication or clustering to ensure data availability and prevent single points of failure.

**Disaster Recovery**

1. Regularly backing up of application code, database, and other critical data to a secure off site location.
2. Implement a disaster recovery plan outlining steps to restore services in case of major outages or infrastructure failures. This may involve deploying from backups to a secondary environment.
3. Regularly test of disaster recovery plan to ensure its effectiveness.

**Test Plan for University Nomination System**

**Testing Objectives:**

**Functionality**: Ensure the system operates as intended, meeting all functional requirements.

**Data Security and Privacy**:Test for vulnerabilities and ensure user data and system integrity are protected..Ensure data privacy and protection regulations are upheld.

**User Experience**: Verify the system is user-friendly, intuitive, and accessible to all target users.Evaluate the user interface's accessibility,error handling, and overall usability.

**Performance and Scalability**: Test system responsiveness and scalability under various workloads.

**Test Scope**

This test plan focuses on:

* User accounts and access control
* Nomination submission process
* Candidate information management
* Review and selection process (if applicable)
* System security and performance
* User interface (UI) and usability

**Testing Approach**

We will employ a multi-phased approach:

1.**Documentation Review**: Proper and careful examination of system requirements and design documents in order to understand its functionalities and purpose better.

2.**Test Planning**: Develop detailed test cases and scripts according to system functionalities and requirements.

3.**Test Environment Setup**: The hardware, software, network, and security settings in the test environment are configured to mirror real-world conditions.

4.**Test Execution**: Systematic execution of test cases.

5.**Reporting**: Documentation of failed test cases with bug descriptions and recommendations for improvement in clear details.

6.**Defect & Regression Testing**: Retesting after fixing issues to validate fixes and ensure no other parts of the system have been affected by the modifications.

7.**Performance & Security Testing**: Specialized testing to assess system resilience to security threats and performance under varying loads.

8.**User Acceptance Testing (UAT)**: Involves testing end to end processes to ensure system meets real-world scenarios and requirements.

**Test Data**

Specify the type of data needed for testing, including:

* Sample user accounts with different roles
* Nomination information with varying scenarios
* Reference data with diverse formats

**Test Cases**

A)**User Accounts and Access Control**

* Test user registration, login, and logout functionalities.
* Verify different user roles and their corresponding access levels.
* Test password reset functionality.

B) **Nomination Submission Process**

* Test successful and unsuccessful submission of nomination forms.
* Verify the system captures all required information correctly.
* Test uploading supporting documents and ensuring file size and type restrictions are applied.

C)**Candidate Information Management**

* Test searching for existing nominees and viewing their information.
* Verify editing and updating nominee information.
* Test the system's handling of duplicate nominations.

D)**Review and Selection Process (if applicable)**

* Test access control for reviewers and selection committee members.
* Verify reviewers can view assigned nominations and submit their feedback.
* Test the functionality of committee discussions and voting (if applicable).

E)**System Security and Performance**

* Test the system for any vulnerabilities against unauthorized access, data breaches, and SQL injection attacks.
* Assess system performance under various load conditions, simulating real-world usage.

F)**User Interface (UI) and Usability**

* Test the user interface for clarity, intuitiveness, and ease of navigation.
* Identify any accessibility issues for users with disabilities.
* Conduct user testing sessions to gather feedback on the overall user experience.

**REFERENCES**

Fernández, E. B., La Red, D., & Peláez, J. I. (2013). A conceptual approach to secure electronic elections based on patterns. *Government Information Quarterly*, *30*(1), 64–73. <https://doi.org/10.1016/j.giq.2012.08.001>

*Essentials - Introduction to software development and testing*. (2024, March 28). Ministry of Testing. <https://www.ministryoftesting.com/courses/essentials-introduction-to-software-development-and-testing>

Grinberg, M. (2014). *Flask Web Development: Developing Web Applications with Python*. <https://dl.acm.org/citation.cfm?id=2621997>

Kaguru, S. (2018). *Design and Implementation of a web-based University Voting Sytem*. GRIN Verlag.

Maia, I. (2015). *Building Web Applications with Flask*. Packt Publishing Ltd.

*IBM Developer*. (n.d.). <https://developer.ibm.com/docloud/documentation>

*Software testing help*. (n.d.). Software Testing Help. https://www.softwaretestinghelp.com/