

**ST. PAUL’S UNIVERSITY**

**SCHOOL OF COMMUNICATION AND COMPUTER STUDIES**

**UNIVERSITY COUNCIL STUDENT CAMPUS NOMINATIONS WEB APPLICATION**

**BY**

**GROUP**

**Q**

**BSCLMR158423**

**BSCLMR573822**

**BOBITLMR735719**

# **PROJECT PLAN**

## **Objective**

To be able to facilitate the nomination process for student campus elections by developing a user-friendly system solution for the University Council.

## **Project Timeline**

1. Planning Phase (Week 1-2):

* Defining project scope, objectives and deliverables.
* Gathering Requirements from stakeholders.

1. Development Phase (Week 3-6):

* Design software architecture and database schema
* Develop fronted and backend components.
* Implement system modeling and integrate with requirements.

1. Testing Phase (Week 7-8):

* Conduct unit testing, integration testing and system testing.
* Fix bugs and optimize performance.
* Prepare user documentation and training materials.

1. Deployment Phase (Week 9):

* Deploy the software on university servers.
* Conduct user acceptance testing.
* Address any last-minute issues and finalize deployment.

1. Post-Deployment Phase (Week 10):

* Provide ongoing support and maintenance.
* Gather feedback for future enhancements.

# **GANTT CHART**

|  |  |
| --- | --- |
| WEEK | ACTIVITY |
| Week 1-2 | Planning Phase  - Defining project scope, objectives and deliverables.  - Gathering Requirements from stakeholders. |
| Week 3-6 | Development Phase   * Design software architecture and database schema * Develop fronted and backend components. * Implement system modeling and integrate with requirements. |
| Week 7-8 | Testing Phase   * Conduct unit testing, integration testing and system testing. * Fix bugs and optimize performance. * Prepare user documentation and training materials. |
| Week 9 | Deployment Phase   * Deploy the software on university servers. * Conduct user acceptance testing. * Address any last-minute issues and finalize deployment. |
| Week 10 | Post-Deployment Phase   * Provide ongoing support and maintenance. * Gather feedback for future enhancements. |

# **SOFTWARE REQUIREMENTS SPECIFICATION (SRS) DOCUMENT**

## **Introduction**

The University Council Nomination System is a web-based application designed to streamline the process of student nominations for campus elections.

## **FUNCTIONAL REQUIREMENTS**

1. User Registration

* Allow students to register with their university credentials.
* Verify student information with the university database.

1. Nomination Process

* Provide a form for students to nominate themselves or others for various positions.
* Validate nominations based on eligibility criteria.
* Allow students to withdraw nominations within a specified timeframe.

1. Voting System

* Implement a secure and anonymous voting mechanism.
* Generate unique voting tokens for each student.
* Ensure one-person-one-vote principle.

1. Admin Panel

* Grant access to university council members for managing nominations and elections.
* Enable admin to view and validate nominations, manage candidate profiles and monitor voting progress.

## **NON-FUNCTIONAL REQUIREMENTS**

1. Security

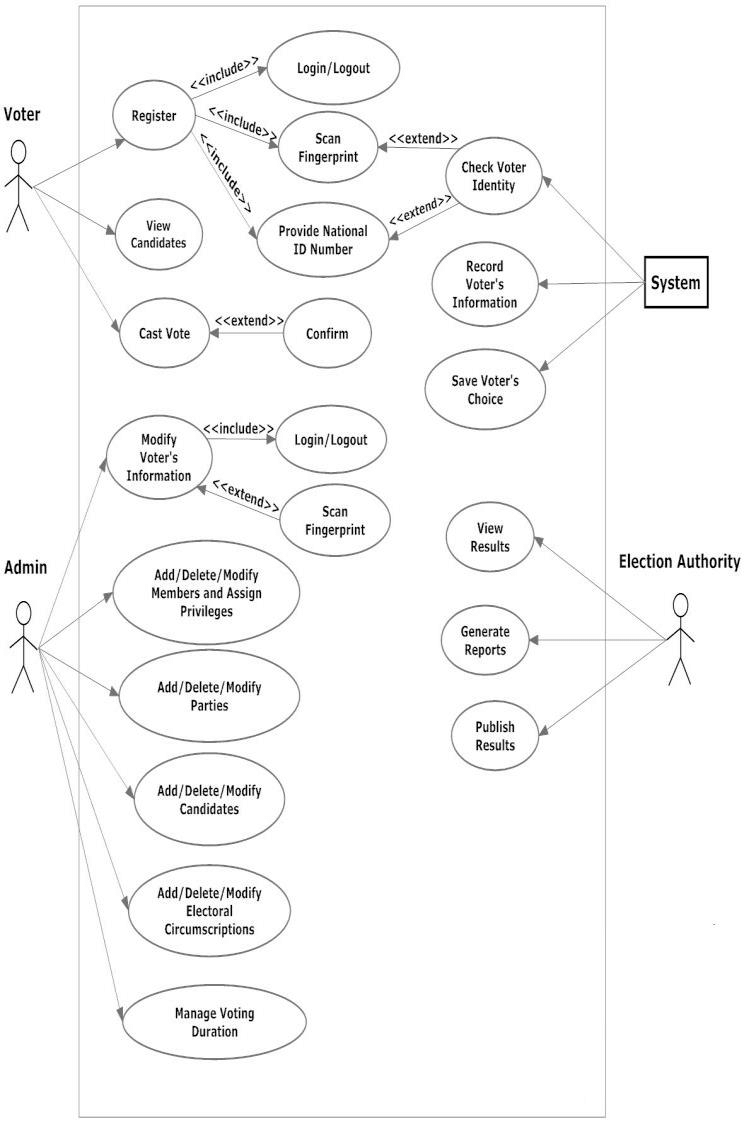
* Implement robust authentication and authorization mechanisms.
* Encrypt sensitive data such as student information and voting records.

1. Scalability

* Design the system to handle a large number of concurrent users during peak times such as election days.

## **SYSTEM MODELING**

Use case diagram



# **REQUIREMENTS TRACEABILITY MATRIX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement ID** | **Requirement Description** | **Design** | **Implementation** | **Testing** | **Deployment** |
| R1 | User should be able to register | Design Document | Implemented | Tested | Deployed |
| R2 | User authentication | Design Document | Implemented | Tested | Deployed |
| R3 | Administrator can create a ballot | Design Document | Implemented | Tested | Deployed |
| R4 | User can cast vote anonymously | Design Document | Implemented | Tested | Deployed |
| R5 | Election results tabulated accurately | Design Document | Implemented | Tested | Deployed |
| R6 | Data encryption for security | Design Document | Implemented | Tested | Deployed |
| R7 | User-friendly interface | Design Document | Implemented | Tested | Deployed |
| R8 | Error messages are informative | Design Document | Implemented | Tested | Deployed |

# **REFERENCES**

1. Alvarez, J., & Beale, J. (2008). Software Requirements. Microsoft Press.
2. Pressman, R. S (2014). Software Engineering: A practitioner’s Approach (8th ed.). McGraw-Hill Education.
3. Sommerville, I. (2015). Software Engineering (10th ed.). Pearson.

https://github.com/ndinitash58/SoftwareEnginneringBCS3106/invitations