**REPORT :**

**Introduction:**

This report gives an overview of my research findings on the database system that was used in UCU main campus Guild Election; and basing on that information to build a similar database system according to my assignment.

Data Sources:

* Mr. Jordan, a worker at the electoral Office
* UCU Online Platform
* Emmanuel Kalyegira, my classmate who participated in the election process till the end.

Data Collection methods:

* Recording
* Interviewing

Data Collection Tools used:

* Phone for recording
* Pen and Book

Findings:

* From Emma, he mentioned that they manually submitted their applications and he mentioned that in order to appear in the nomination list, your application has to be approved. Your not informed of your rejection, you just wait for the nomination list to know your fate.
* Mr. Jordan couldn’t give the Entities he used in the recent 2024 Guild Election, because he said its upon us to come up with them ourselves depending on the business requirements, we need or the data we want to record and save. The 2024 elections were based on functional (what a system should do) and non-functional requirements (how the system should perform). There are different types of elections that were carried out namely; Student Election, Guild election, International Students association etc.

In a basic election, there are dates (opening and closing dates of all the processes). There is also configuration of constituencies according to hierarchy starting from the university level, campus level, faculty, departments under the faculties, programs in that department. Constituencies should have attributes and datatypes like null, not null.

About positions(posts): These positions are attached to a consistuency but there is a core position table having attribute like PoitionID, Position name, Constituency(where it should belong)

Problems / gaps identified:

I was not able to get this detail but based on my assumption, these might be the problems and gaps as seen below;

* Inadequate validation mechanisms may lead to duplicate voting of candidates
* A person cannot undo a vote once submitted.
* Manual submissions of the application by candidates.

Functional data requirements identified:

* There is secure management of voters, candidates and election processes.
* There is real-time tracking of votes, nominations and election types
* There is role-based access control for admins and voters
* The system allows users to vote for candidates

Uses a check constraint to ensure no duplication of votes.

* The system tallies the votes of the candidates’ votes and ranks them according to the number of votes they have received.

**Logical Assumptions from my findings:**

* Every voter has a unique email.
* Only candidates whose applications were approved participated in the nomination process.
* Every election process has a unique ID.
* Administrators check eligibility of candidates before vetting or rejecting a candidate in the vetting process.
* Each user has a unique ID to differentiate them within the system.

All candidates must be approved through an application and vetting process before being added to the official candidates' list.

* Votes submitted are final and cannot be reversed or modified
* Only authorized voters can vote for specific posts.
* The system has real-time validation to ensure invalid operations show an error.
* Students cannot access sensitive information like who voted who or who nominated who among others.

Rules Governing the Election Process:

* The students or user table contains their credentials which they must use when logging in.

A check constraint to see if a person has already voted.

* Students can only apply for posts depending on their level of access (i.e. Positions under a faculty)
* Voters must be registered in the system, those not registered cannot vote.
* Residency and nationality requirements must be satisfied to participate in specific election posts.
* Clear timelines are set for applications, nominations, vetting, voting, and result declaration.
* Each Electoral member is assigned a role to carry in the election process.
* Votes are tallied immediately and candidates get ranked depending on the number of votes they received.
* Criteria like Residency status, faculty of a student or voter and their nationality must

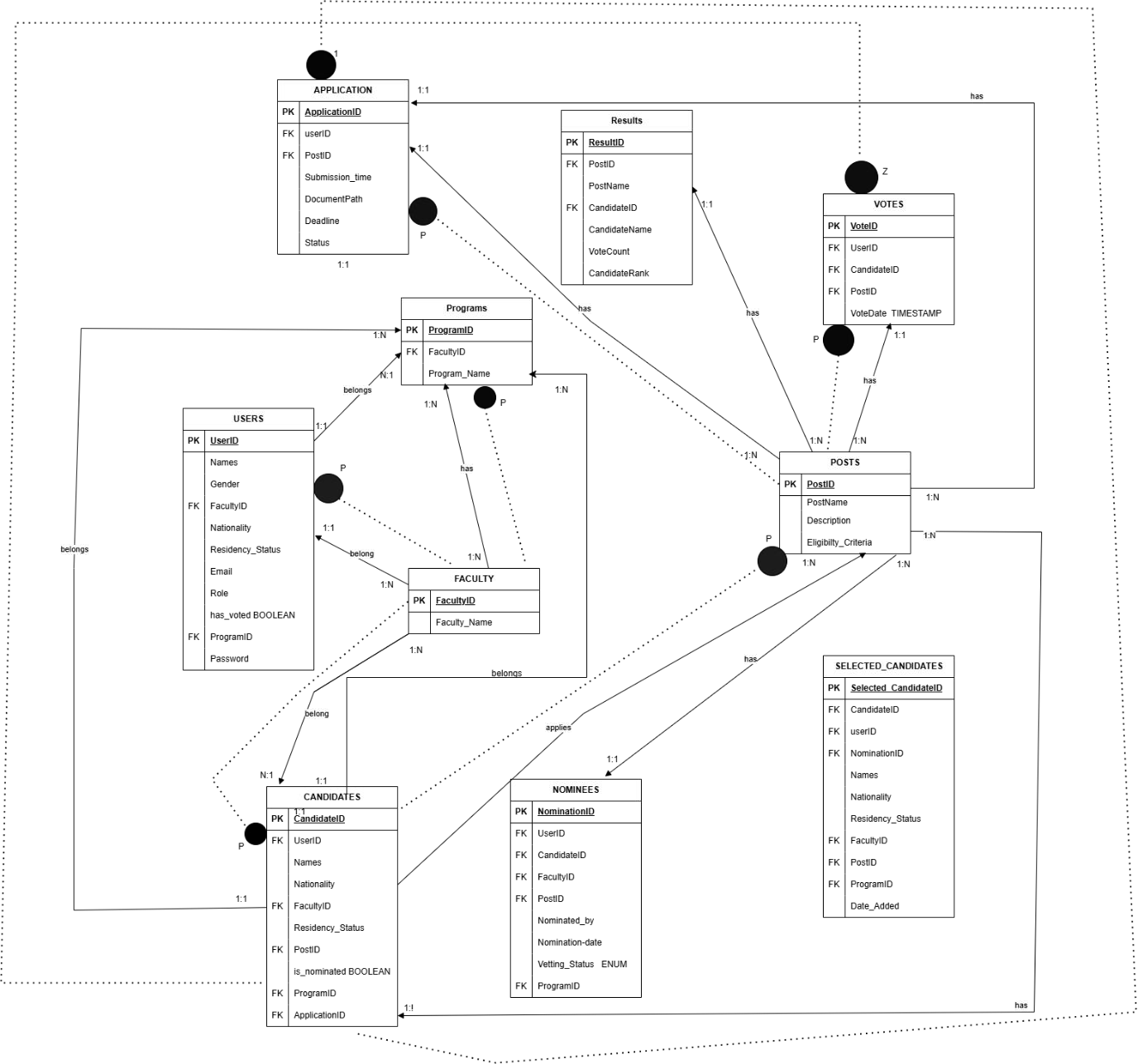
be satisfied before participation in an election process.

* Each user may nominate and vote two candidates per post.
* One Candidate, one vote in a post

Enhanced Entity Relationship Diagram for UCU\_Electoral\_System using IDE-FIX notation.

Strong Entites:

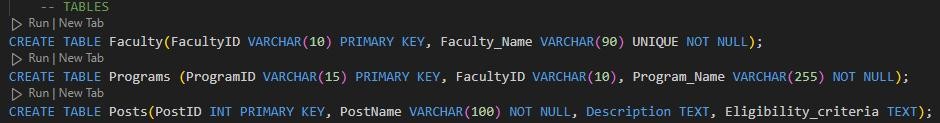
* Faculty
* Posts



**DATABASE STRUCTURE** Table in My Database:

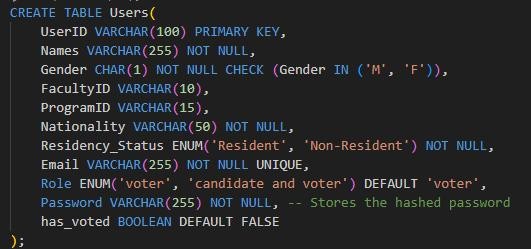
I came up with the following tables;  Faculty table: This stores information about the different faculties belonging in the university.

* Programs: This stores information about the programs offered by the faculties.
* Posts Table: Information about the posts available for candidates to apply for.

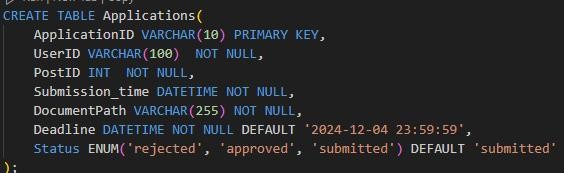




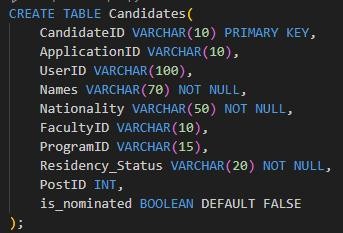
* Users table: This basically shows the users of the system, storing their credentials and passwords and also their role in the the election process.



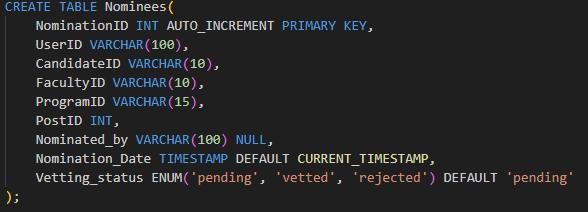
* Application table: Information about applications submitted before deadline and there status. Ie. Whether approved, pending or rejected. Those that are approved(meaning they met the criteria) automatically become candidates and are transferred into the candidates table.



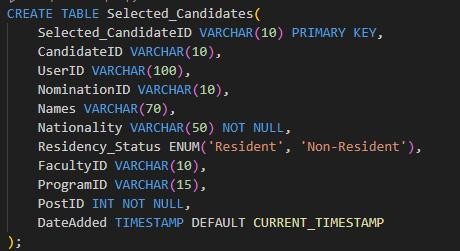
* Candidates table: Just details about candidates who passed the application table and are now up for nomination. They are nominated from this table, those that are nominated by users have their nomination status updated to true and automatically transferred into the Nominees table.



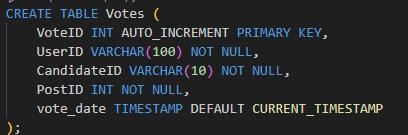
* Nominees table: Nominations by users are tracked here, (who nominated who). Those who passed through the two processes (application and nomination) are then vetted and their status is changed by an admin to vetted; after those who are vetted automatically go into the Selected candidate table (this table is present to users to know who they can vote for)



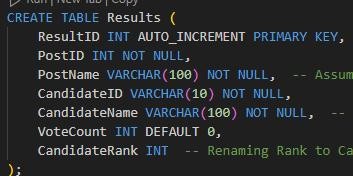
* Selected candidate table: Final list of those eligible to be voted for.



* Votes table: Stores individual votes cast by users. This also triggers a column in the users table(has\_voted) whereby its updates to true whenever a user votes to show they have completed voting.



* Results table: Stored the vote totals for a candidate and ranks them by postID. This table keeps updating whenever a vote is cast.



CONSTRAINTS USED

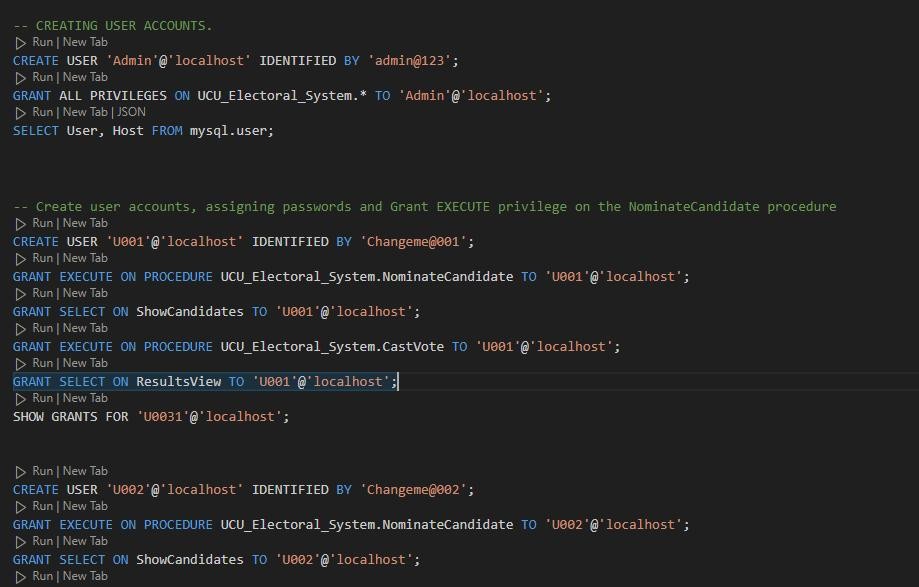
* Check constraint (checks for the @, gender

(M, F))

* Unique constraint
* Null and Not null
* Primary key constraints
* Foreign keys to link tables for referential integrity
* Auto Increment constraint: Generates a unique value for a column often used for primary keys whenever a new record is inserted.
* Default: Provides a default value for a column if no value is specified during an insert operation.

**User Accounts and management**

* Created sample users.ie. an admin with all privileges on the database, admin1 with all privileges except to drop or delete anything and other three users who log in using their credentials granting them permission to execute procedure CastVote and NominateCandidate.
* Users can also select on views for information I want them to know. These user accounts also have access to see specific columns in a specific tables.



Functional and Non-Functional Requirements in my Database (UCU\_ Electoral\_System) Based on my research, I ensured the following requirements on my database.

**1.Functional Requirements (What the system should do)**

# a.Management for Nomination

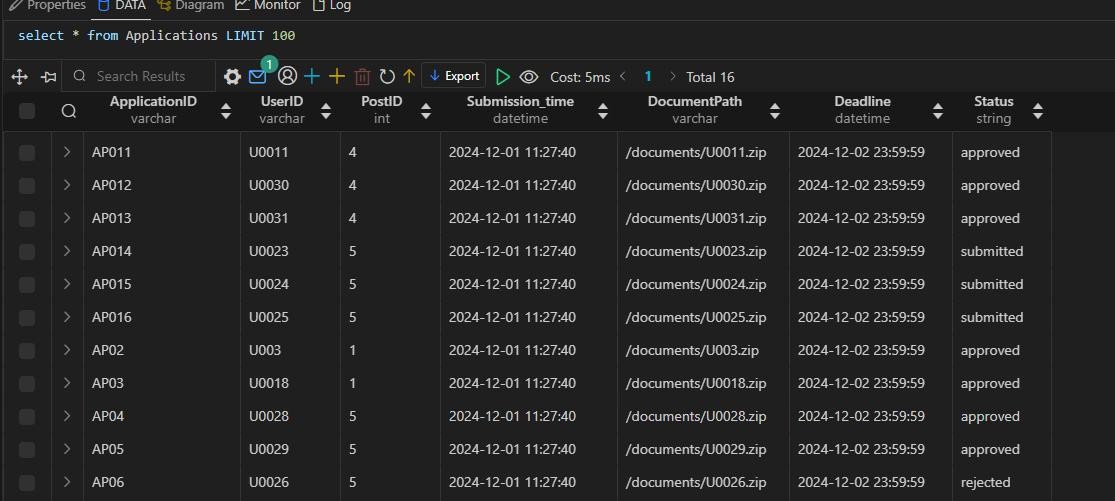
* Users should be able to nominate candidates for a specific post.
* Users are restricted a specific number of nominations for a post.
* Validate Nominations based on eligibity rules (e.g., nationality, residency status).

# b.Management of Users

* Management of user roles( eg. admin, voters and candidate and voter)
* Authentication of users/voters via UserID and Password

# Applications

* Applications are allowed to submit their applications themselves through insertions as long as it not passed deadline.
* There is validation of applications based on submission deadline and document type as .zip.
* In the application status, an admin is able to update the status to(approved, rejected, pending) if the candidate meets the criteria. EG. Admin logged in and approved some of the applications by updating the status.



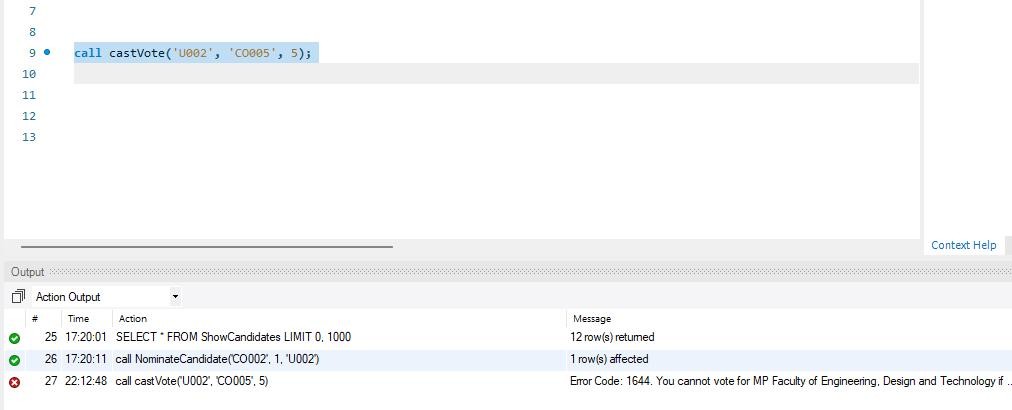
# d.Voting System

* Allows only users in the system to vote for posts
* Allows even Candidates to nominate and vote themselves.
* Users are restricted from voting multiple times and for a candidate.
* Voting is based on the restriction applied to a given post (MP International, MP resident, MP Faculty etc)

EG. U001, a Ugandan user trying to vote for a candidate standing for an MP International post



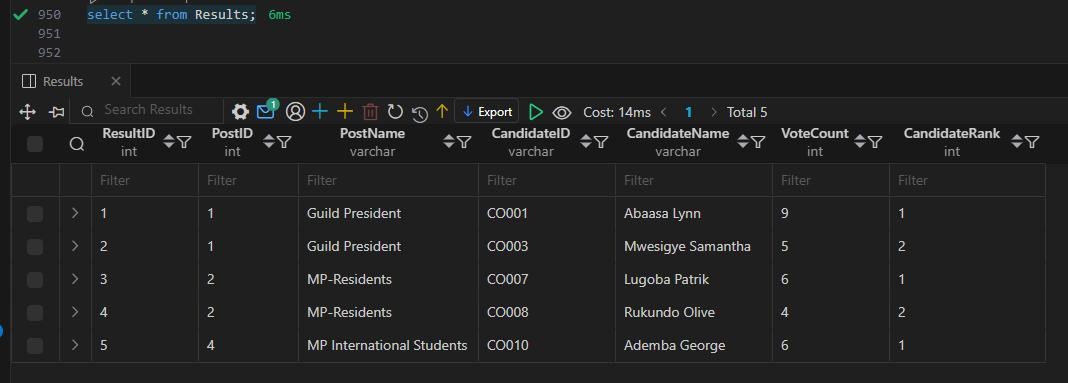
# EG. U002 from Law faculty voting for MP Faculty engineering



# e. Final Election Results

* Ranks candidates based on the votes received from the Vote

table



* A filtered view to show users final results allowing them access to specific columns (ResultsView) when they run a select query on that table.

# f. View for Data Access

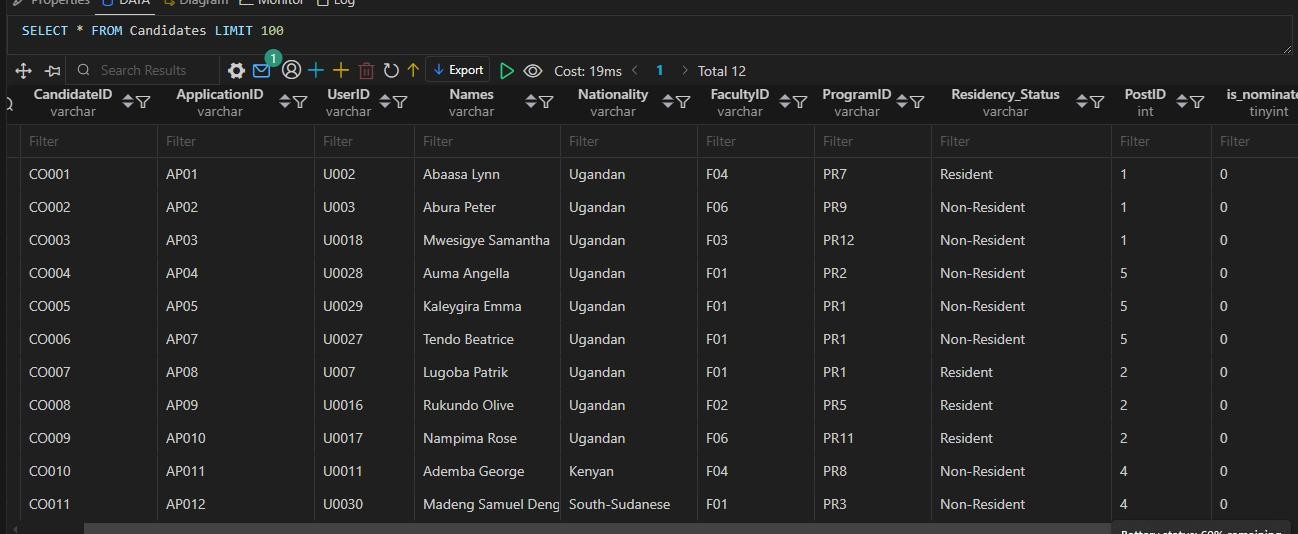
 Provided filtered views I want users to access from specific tables (e.g., ShowCandidates, RejectedApplications, ResultsView).

# g. Audit and Monitoring

 Tracking and logging of actions(voting, nominations).

# h.Post Management

* A candidate can apply for one post
* There is a table Posts that displays all the available posts
* Candidates are linked to their respective faculties, programs and residency\_Status.



i. Triggers

* Used them to enforce system rules and transfer of data to different table.

2.**Non-Functional Requirements** (How the system should perform)

# a)Performance

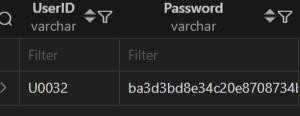
 The system should handle multiple requests, insertions without delays.

# b)Scalabilty and flexibilty

* The system is able to adjust for future adjustments giving allowance for change without affecting the system performance.

c)Security

* Strong password hashing (e.g. SHA256) for users.
* Restricted major operations to the admin



d) Data Integrity:

* Enforced rUsed foreign keys to ensure data integrity between tables
* Ensured Data validity during insertions, updates and deletions.