ERD Presentation Notes: University E-Voting System

These notes are designed to help you explain the Entity Relationship Diagram (ERD) during your presentation. They focus on the **Cardinalities** (how many items relate to one another) and **Multiplicities** (constraints on the relationship).

---

1. High-Level Overview

- **Database Engine:** MySQL
- **ORM:** Prisma (for type-safe database interactions)
- **Core Logic:** The database separates **Authentication** (Users) from **Voting Eligibility** (EligibleVoters) to ensure security and scalability.

---

2. Key Entities & Relationships (Detailed Breakdown)

A. User & Candidate Relationship

- **Entities:** `User` (Logins) and `Candidate` (Nomination Profiles).
- **Cardinality: One-to-Many (1:N)**
  - **Explanation:** A single `User` (with the role CANDIDATE) can technically create multiple `Candidate` records over time (e.g., running for different positions in different years).
  - **Multiplicity:**
    - **User side:** `||` (Mandatory One) - A Candidate profile *must* belong to a User.
    - **Candidate side:** `o{` (Optional Many) - A User *might* not have any candidate profiles yet (if they just registered), or they could have multiple.

B. Position & Candidate Relationship

- **Entities:** `Position` (e.g., "Guild President") and `Candidate`.
- **Cardinality: One-to-Many (1:N)**
  - **Explanation:** One `Position` (like "President") has many `Candidates` competing for it.
  - **Multiplicity:**
    - **Position side:** `||` (Mandatory One) - A Candidate *must* run for a specific Position.
    - **Candidate side:** `o{` (Optional Many) - A Position *can* have many candidates (or zero if nominations haven't started).

C. EligibleVoter & Verification (OTP)

- **Entities:** `EligibleVoter` (The student registry) and `Verification` (OTP codes).
- **Cardinality: One-to-Many (1:N)**
  - **Explanation:** One `EligibleVoter` can request multiple OTPs.
  - **Why?** If an OTP expires or is lost, the voter needs to request a new one. We track all requests for security auditing.
  - **Multiplicity:**
    - **Voter side:** `||` (Mandatory One) - An OTP *must* belong to a registered voter.
    - **Verification side:** `o{` (Optional Many) - A voter might not have requested an OTP yet.

D. Verification & Ballot

- **Entities:** `Verification` (Successful OTP check) and `Ballot` (The "ticket" to vote).

- **Cardinality: One-to-One (1:1)**
  - **Explanation:** A single *successful* verification event generates exactly **one** Ballot token.
  - **Multiplicity:**
    - **Verification side:** `||` (Mandatory One) - A Ballot *must* come from a verification.
    - **Ballot side:** `||` (Mandatory One) - A successful verification yields exactly one ballot.

E. Ballot & Vote

- **Entities:** `Ballot` and `Vote` (The actual choice cast).
- **Cardinality: One-to-Many (1:N)**
  - **Explanation:** A single `Ballot` contains multiple `Votes`.
  - **Why?** A ballot allows the user to vote for multiple positions at once (e.g., 1 vote for President, 1 vote for GRC, 1 vote for Treasurer).
  - **Multiplicity:**
    - **Ballot side:** `||` (Mandatory One) - A Vote *must* be linked to a valid Ballot.
    - **Vote side:** `|{` (Mandatory Many) - A Ballot *must* contain at least one vote to be considered "cast" (or usually all votes are submitted together).

---

3. Cardinality Notation Guide (Crow's Foot)

If you are showing the diagram, explain the symbols at the ends of the lines:

- `||` **(Double Dash): Mandatory One**.
  - *Example:* A `Vote` must belong to exactly one `Ballot`.
- `|{` **(Dash and Crow's Foot): Mandatory Many** (One or More).
  - *Example:* A `Ballot` contains one or more `Votes`.
- `o|` **(Circle and Dash): Optional One** (Zero or One).
  - *Example:* A `User` might be a `Candidate` (1), or just a regular Admin (0).
- `o{` **(Circle and Crow's Foot): Optional Many** (Zero or More).
  - *Example:* A `Position` might have zero `Candidates` (if created today) or many `Candidates` (during election week).

---

4. Summary Script for Presentation

"Our database schema is built on **MySQL** and managed via **Prisma ORM**. The core of our design is the separation of concerns:

1. **Users & Roles:** We have a **One-to-Many** relationship between Users and Candidates, allowing flexibility for users to run for office.
2. **Election Structure:** A **One-to-Many** relationship exists between Positions and Candidates—one position, many contestants.
3. **Secure Voting Flow:**
   - We use a **One-to-Many** relationship for OTPs, allowing voters to retry verification if needed.
   - Once verified, we enforce a strict **One-to-One** relationship to generate a unique, single-use Ballot token.
   - Finally, that single Ballot links to multiple Votes (**One-to-Many**) covering all the positions in the election.

This structure ensures **Data Integrity** (no duplicate votes) and **Auditability** (every action is traceable)."