1. Introduction:
    - Objectives - Create a functional vote collection and tabulation system taking data from both mobile and desktop applications and websites.
    - Major Functions -
        - Accept voting data from applications and websites coming from mobile and desktop devices
        - Store the ballot collection and tabulation policy
        - Receive the ballots
        - Calculate vote totals regardless of First Past the Post (FPTP) or Ranked voting systems
        - Report the results
        - Generate a correctness proof for the vote totals
    - Deliverables -
        - Tabulation and weighting system.
            - Database Access subsystem
        - Project presentation
        - Term paper
        - Requirements Analysis
        - System Design
        - Object Design
        - Test Plan
    - Evolution of the Software Project Management Plan:
        - The software project management plan is under version control. Proposed changes and new versions of the plan will be posted to the group discord and will be available to all the project members.
    - Management and Technical Constraints
        - Deadline of Nov 29th (Project Presentation)
        - Limited experience with project management and development
2. Project Estimates:
    - Historical Data Used - Using a bottom-up approach we are going to be estimating time, based on previous school project timelines.
    - Estimation Techniques Used - Bottom-up time estimation technique
    - Resource and Project Duration Estimates - Will be done in C++ using Visual Studio, code will be backed up in GitHub. The project was started on Aug 26, 2022, and is planned to end on Dec 04, 2022.
3. Schedule:
    - Work Breakdown Structure

Task Name	Assigned Role	Description	Task Input	Task Output
Database	System	Elicit requirements	Team Liaisons	Analysis of

Access Subsystem requirements elicitation	Architect & Developers	for the access system from the voting teams		database
Database Access Subsystem design	System Architect & Developers	Design the subsystem responsible for retrieving data from the voting databases	Analysis of database	Database Access Subsystem design
Database Access Subsystem implementation	Developer	Implement the access subsystem design	Database Access Subsystem design	Database access subsystem source code
Database Access Subsystem Inspection	Developer	Test the database access subsystem implementation	Database access subsystem source code	List of discovered defects
Database Access Subsystem test plan	Tester	Develops a test plan for testing the Database access subsystem	Database access subsystem source code	Test and test plan
Database Access Subsystem test	Tester	Execute the test plan	Database access subsystem source code	Test results, list of defects
Vote counting / weighting subsystem requirements elicitation	Domain Specialist	Elicit from the customer the requirements for the vote counting / weighting subsystem	Customer	Vote counting / weighting subsystem requirements
Vote counting / weighting subsystem design	System Architect	Design subsystem responsible for counting votes; capable of handling FPTP and Ranked voting systems	Vote counting / weighting subsystem requirements	Vote counting / weighting subsystem design
Vote counting / weighting subsystem implementation	Developer	Implement vote counting / weighting subsystem design	Vote counting / weighting subsystem design	Vote counting / weighting subsystem design
Vote counting / weighting	Developer	Test the vote counting / weighting	Vote counting / weighting	List of discovered

subsystem Inspection		subsystem implementation	subsystem source code	defects
Vote counting / weighting subsystem test plan	Tester	Develops a test plan for testing the vote counting / weighting subsystem	Vote counting / weighting subsystem source code	Test and test plan
Vote counting / weighting subsystem test	Tester	Execute the test plan	Vote counting / weighting subsystem source code	Test results, list of defects

- Gantt Chart Representation - Attached as a PDF

4. Project Resources:
   - People - Marissa Morton, Miguel Gapud, Abram Miller, Marissa Morton, Michael Zuppardo, Alexander Mcnair
   - Hardware and Software - visual studio, visual paradigm, GitHub
   - Special Resources - SQL Database (for stored voting data)
5. Staff Organization:
   - Team Structure
     - Marissa Morton - 1st Project Manager / Document Editor / Developer / Liaison / System Architect
     - Miguel Gapud - Developer / Assistant Configuration Manager / Tester / System Architect
     - Abram Miller - Developer / Configuration Manager / Tester
     - Alexander Mcnair - Liaison / Domain Specialist
     - Michael Zuppardo - Domain Specialist
   - Management Reporting - Weekly status meetings / Regular reviews
6. Risk Management Plan:
   - Risk Analysis - Incomplete list of the many roadblocks that may occur.
     - External Servers crashing
     - Local data loss
     - Participant(s) becomes unable to contribute for any reason(s)
     - Illnesses or injuries
     - Code breaking between platforms (macOS vs ios vs Windows vs Linux)
     - Changes in other groups affecting our group
     - Natural disasters affecting Shelby Hall or the university in a way that everyone is sent home and cannot meet in person
     - Buggy release
     - Change in professor
   - Risk Identification - Issues that are devastating or likely enough to account for

1. A participant dropping the class and leaving early or becoming otherwise able to contribute to the project in any way.
2. Groups changing how they want to store their voting information.
3. Local data loss
4. Illnesses and injuries (including covid)
5. Code breaking between platforms (macOS vs ios vs Windows vs Linux)

- Risk Estimation - How likely are risks and how devastating are they
  - Devastation - Loss of Hours, Days, Weeks, Entire Project

1. Loss of a participant - ~30%, Days
    a. Loss of 2 participants - ~10%, Weeks
    b. Loss of 3+ participants - <5%, Entire Project
2. Groups changing how they want to store voting information - 90%, Hours
3. Local data loss - 60%, Days to Weeks depending on the data
4. Illnesses and injuries - 50%, Hours: 10%, Days
5. Code breaking between platforms (macOS vs ios vs Windows vs Linux)

- Risk Abatement Procedures

1. Loss of a participant - Make sure important jobs are not solely handled by one person. If someone were to become unable to contribute, make sure there is someone else similarly capable of picking up their project.
2. Groups changing how they want to store voting information - Delay work on code accessing other teams' databases until the other team has solidified their decision on how that data will be stored.
3. Local data loss - Make sure nothing is stored on one computer alone. Either make two team members store redundant data, or make sure it is stored both locally and on the cloud because one is none; two is one.
4. Illnesses and injuries - For lighter illnesses and injuries where work is still possible, make sure remote work is possible. Zoom and Discord calling are both avenues to mitigate the problem. For more serious illnesses and injuries, we may need to use the Loss of a participant abatement process.
5. Code breaking down between platforms - Look through testimonies of other software developers with experience creating software that accesses websites and applications. Be sure to minimize platform-specific segments of code if they must exist.

7. Project Tracking and Control Plan

- Developers will log what they have implemented and the purpose for implementing a piece of code. This should be done by writing descriptive comments within the code specifying the intended purpose of each section and a Google Doc containing a summary of what they have implemented and when.
- Liaisons will log their communications with other teams. This will be done in a Google Doc and will include the date of communication, key topics discussed, and any actionable feedback or data for our team.

8. Miscellaneous Plans - plans that will be created but are not part of the SPMP

- Quality Assurance Plan
- Configuration Management Plan
- System Testing Plan