ASCI Senate Voting System: Project Introduction

The College of Idaho Coding Club is developing a customized voting system to meet the Senate Committee's needs while addressing the efficiency and transparency requirements. The voting system will allow senators to securely log in, cast their votes in real time, and view immediate results, with all session data being recorded and exportable. This project is being developed as an independent study course worth 1 credit over a 1-month timeframe, with daily 1-hour meetings (5 days a week). The team of 5-7 students is working on the project under the supervision of Dr. Jonny Comes.

The final deliverables include:

1. The app codebase.
2. A detailed database diagram (CSC 270 / LucidChart).
3. Comprehensive project documentation.

The key technologies are **Node.js** (MVC architecture), **Bootstrap** (frontend templates), and **Socket.io** (real-time voting).

Project Goals:

- Learn and apply full-stack web development principles.
- Design relational database management systems.
- Gain hands-on experience with frontend frameworks and web sockets.
- Build a functional MVP to support Senate voting sessions.

Week 1: Planning & Setup

Goal: Establish project groundwork, allocate responsibilities, and begin development.

- **Day 1**:
  - Finalize project scope and requirements.
  - Discuss feature prioritization and break tasks into modules.
  - Set up a GitHub repository and define the branching strategy (e.g., feature branches).
  - Assign team roles (frontend, backend, database, project manager).
- **Day 2-3**:
  - Create the database schema and generate the LucidChart diagram.

    - Design basic UI wireframes (Bootstrap components for login, dashboard, and voting screen).
- **Day 4-5**:
    - Set up the development environment (Node.js, Bootstrap).
    - Implement user authentication and role-based access control (Secretary and Senators).
    - Integrate basic session creation logic.

Deliverables:

- GitHub repo initialized with README.
- Initial database schema in LucidChart.
- Wireframes for UI.
- Authentication and session creation code.

Week 2: Core Functionalities

Goal: Build the main application logic and backend services.

- **Day 1-2**:
    - Develop attendance recording features (frontend form, backend routes).
    - Create backend routes for adding and managing voting sessions.
- **Day 3-4**:
    - Build voting mechanism (real-time broadcast using Socket.io).
    - Implement role-specific dashboards (Secretary: session control, Senators: voting page).
- **Day 5**:
    - Test the integration of attendance and voting features.

Deliverables:

- Functional attendance recording and session management.
- Real-time voting implemented with Socket.io.
- Working role-specific dashboards.

Week 3: UI, Testing, and Feedback

Goal: Polish frontend design and fix bugs.

- **Day 1-2**:
  - Style the application using Bootstrap templates.
  - Refine responsiveness for different devices.
- **Day 3-4**:
  - Test the app thoroughly (unit testing for backend, usability testing for frontend).
  - Address bugs and edge cases (e.g., session timeouts, duplicate votes).
- **Day 5**:
  - Review feedback from mock users (e.g., senators).

Deliverables:

- Completed and styled UI.
- A list of fixed bugs and improvements.
- Feedback report from mock users.

Week 4: Finalization & Documentation

Goal: Prepare the project for submission and deployment.

- **Day 1**:
  - Document project setup, usage, and API endpoints.
- **Day 2-3**:
  - Add export functionality for session voting records in spreadsheet.
  - Ensure the app supports manual publishing of records.
- **Day 4-5**:
  - Prepare the final submission package (codebase, LucidChart diagram, documentation).
  - Conduct a final project demo.

Deliverables:

- Comprehensive documentation.
- Completed database diagram.
- Fully functional app submitted on GitHub.
- Final project demo presentation.