The Middles

E-Voter

Miles Griffith, Nathan Ross, Vital Kyungu, & Brooks Halliday

02/23/25

SRS Document

Functional Narrative

The completed website will serve as a live video hosting platform allowing citizens to view real-time debates and hearings held in local, state, and federal government offices. Users will be able to log in and access legislative discussions relevant to their location, providing them with a transparent view of lawmaking processes. Additionally, citizens will have the ability to cast votes on legislative matters, expressing their opinion on whether a proposed law should be passed.

As an official government website, it will be fully ADA-compliant, ensuring accessibility for all users, including those with disabilities. The site will feature a dashboard where users can track ongoing discussions, review legislative history, and see which proposals have been approved or rejected. The platform will prioritize security, user privacy, and seamless navigation to encourage civic engagement and informed decision-making.

The prototype will be a simplified version of the full web app, focusing on core functionalities without implementing login/registration or real-time video streaming. Instead of hosting live sessions, it will provide pre-recorded videos representing legislative debates and hearings. Users will be able to vote on the topics presented in the videos and submit feedback or suggestions for website improvements, which will be stored in a local database. The prototype will retrieve and display pre-recorded legislative videos stored in a database. Videos will be listed on a dedicated page, and users will be able to select and play them.

To simulate the location-based services planned for the full version, the prototype will include an API-driven ZIP code lookup, allowing users to enter their ZIP code and receive information about their corresponding state or county. This approach will help demonstrate how localized legislative content will be structured in the final implementation. The prototype will serve as a proof of concept, focusing on usability, engagement, and refining the core user experience before full-scale development.

System Scope

The full system will encompass a comprehensive live video hosting platform that facilitates government transparency by broadcasting real-time legislative debates and hearings at the local, state, and federal levels. It will incorporate user authentication, allowing citizens to securely log in and access discussions relevant to their jurisdiction. Users will be able to vote on proposed legislation, track decisions through a historical dashboard, and engage with an ADA-compliant, fully accessible interface. The system will require real-time video streaming infrastructure, high-availability servers, robust security measures, and scalable architecture to support large volumes of users and interactions. Additionally, it will integrate location-based services, automatically tailoring content to users based on their geographic region.

Project Scope

The prototype will focus on demonstrating the system's core functionality in a simplified environment. Instead of live video streaming, it will host pre-recorded videos related to legislative topics. Users will be able to vote on video topics and submit feedback about the platform, with responses stored in a local database. The prototype includes a real-time ZIP code lookup feature that retrieves and displays state and city data from an external API. This initial version will be used for testing user experience, and interface design.

Assumptions

1. *Platform and Environment:*
    - The application will be a web-based platform accessible via modern browsers (e.g., Chrome, Firefox, Safari, Edge).
    - The app will be hosted on a cloud-based infrastructure to ensure scalability and reliability.
    - The system will support both desktop and mobile-responsive designs for optimal user experience across devices.
2. *Compliance:*
    - The application will comply with relevant data protection regulations (e.g., GDPR, CCPA) to ensure user privacy and data security.
    - The app will adhere to accessibility standards (e.g., WCAG 2.1) to ensure inclusivity for users with disabilities.
3. *Constraints:*
    - The system will rely on third-party APIs (e.g., live-streaming services, translation services) for certain functionalities, which may introduce latency or dependency issues.

- The advisory voting system will not be legally binding and is intended for informational and engagement purposes only.
- The app will require users to create an account to participate in voting and provide feedback, which may limit anonymous participation.

4. *Technical Assumptions:*
   -
   - Real-time updates (e.g., live streams, vote tallies) will require a robust backend infrastructure to handle high traffic and ensure low latency.
   - Captions and translations for live-streamed content will be generated using AI-powered tools, which may have occasional inaccuracies.
   - The Prototype will be implemented using the sails.js framework
   - The prototype's database will be Postgresql
5. *User Assumptions:*
   - Users will have basic familiarity with web applications and navigating dashboards.
   - Users will have access to a stable internet connection to participate in live-streamed debates and cast votes.
   - Users will provide constructive feedback to help improve the platform's functionality and usability.
6. *Administrative Assumptions:*
   - Administrators will have access to a dedicated dashboard for monitoring user engagement, reviewing feedback, and generating reports.
   - Feedback and issue reports will be addressed within a defined timeframe (e.g., 48 hours) to maintain user trust and satisfaction.
   - Aggregated voting data will be anonymized to protect user privacy while still providing actionable insights to representatives.
7. *Content Assumptions:*
   - Legislative content (e.g., bills, debates) will be provided by trusted sources and updated in real-time.
   - Live-streamed debates will include captions and translations in multiple languages to cater to a diverse user base.
   - The app will provide a curated selection of sample votes and videos for demonstration purposes during the prototype phase.
8. *Feedback System Assumptions:*
   - The feedback system will include a structured form to capture user input effectively.
   - Feedback submissions will be categorized (e.g., bug reports, feature requests) to streamline the resolution process.
   - Users will receive notifications about the status of their feedback submissions (e.g., received, in progress, resolved).

9. *Scalability Assumptions:*
    - The system will be designed to handle a growing number of users and legislative items as the platform expands.
    - The backend infrastructure will support horizontal scaling to accommodate increased demand during peak usage periods (e.g., major legislative events).
10. *Security Assumptions:*
    - The app will implement encryption (e.g., HTTPS, TLS) to secure data transmission between the user's device and the server.
    - User data (e.g., votes, feedback) will be stored in a secure database with restricted access to authorized personnel only.
    - Feedback submissions will be first validated to ensure no malicious or empty contect is submitted.
    - Regular security audits will be conducted to identify and address potential vulnerabilities.

Functional Requirements (FR)

ID	Functional Requirement	User Story	Test case	Use Case
FR-1	Users can watch pre-recorded videos	US-1	T-1	USC-1
FR-2	Users can vote on topics after watching videos	US-2	T-2	USC-2
FR-3	Users can submit feedback on the platform	US-3	T-3	USC-3
FR-4	Users can enter a zip code or address and be given the relevant location data	US-4	T-4	USC-4

List of Steps

FR-1. Watching Pre-recorded Legislative Videos

FR-1.1. User visits the website
FR-1.2. Homepage displays a list of pre-recorded legislative videos.
FR-1.3. User selects a video.
FR-1.4. Video plays with basic playback controls (play, pause, volume).

FR-2. Voting on Video Topics

FR-2.1. User watches a video.
FR-2.2. After watching, a voting interface appears.
FR-2.3. User selects a voting option (e.g., "Support" or "Oppose").

FR-2.4. Vote is stored in the local database.
FR-2.5. A confirmation message appears.

FR-3. Submitting Feedback and Suggestions

FR-3.1. User clicks on a "Submit Feedback" button.
FR-3.2. A feedback form appears.
FR-3.3. User enters comments or suggestions.
FR-3.4. User submits the form.
FR-3.5. Feedback is stored in a local database.
FR-3.6. A confirmation message appears.

FR-4. Entering ZIP Code for Location Lookup

FR-4.1. User visits the location section of the site.
FR-4.2. A prompt asks the user to enter a ZIP code.
FR-4.3. User inputs a ZIP code.
FR-4.4. The system uses an API to determine the corresponding state or county.
FR-4.5. The result is displayed on the page.

Non-Functional Requirements (NFR)

NFR-1: Usability Requirements

NFR-1.1: The user interface shall be designed to allow a user to complete a voting action or submit feedback within 3 clicks from the homepage.

NFR-1.2: The system shall comply with WCAG 2.1 accessibility guidelines, ensuring compatibility with screen readers and keyboard navigation.

NFR-1.3: The website shall support responsive design, ensuring usability on mobile devices with a minimum resolution of 375x667 pixels.

NFR-2: Compatibility Requirements

NFR-2.1: The prototype shall function correctly on Chrome, Firefox, Edge, and Safari, tested on their latest stable versions.

NFR-2.2: The system shall support both desktop and mobile devices, ensuring proper scaling and interaction.

NFR-3: Maintainability Requirements

NFR-3.1: The system shall be designed using modular programming principles to facilitate future enhancements and maintenance.

NFR-3.2: API calls for ZIP code lookup shall be structured for easy replacement or scaling in the final version.

NFR-3.3: The system shall generate error logs for all major system failures to assist with debugging and future improvements.

NFR-4: Data Storage & Integrity Requirements

NFR-4.1: The system shall confirm successful storage of votes and feedback within 2 seconds after submission.

NFR-4.2: The ZIP code lookup API shall return a valid response at least 95% of the time.

NFR-4.3: The system shall store votes and feedback persistently so that data is not lost on refresh or session timeout.

User Stories (US)

ID	User Story
US-1	As a user, I want to watch pre-recorded legislative videos, so that I can stay informed on government discussions.
US-2	As a user, I want to vote on the topic of the video, so that I can express my opinion on the legislative matter.
US-3	As a user, I want to submit feedback and suggestions about the platform, so that I can contribute to its improvement.
US-4	As a user, I want to enter my ZIP code and see my corresponding state or county, so that I can understand how location-based filtering would work in the full system.

Acceptance Tests (T)

Test ID	Test Scenario	Expected Outcome	Pass/Fail Criteria	Matching Requirement
T-1	User selects and plays a video	Video plays with playback controls	Pass if the video loads and plays	FR-1
T-2	User votes on a video topic	Vote is stored, and confirmation appears	Pass if vote is recorded and confirmation is displayed	FR-2

T-3	User enters and submits feedback	Feedback is stored, and confirmation appears	Pass if feedback is recorded and confirmed	FR-3
T-4	User enters a ZIP code	The system shows an error message	Pass if location matches expected result	FR-4

Use case Scenario (UCS)

UCS-1: Watching a Legislative Video

- Actor: Website Visitor
- Precondition: The user has accessed the website.
- Steps:

    UCS-1.1: User selects a video from the homepage.

    UCS-1.2: The video player loads.

    UCS-1.3: User clicks play, pause, or adjusts volume.

- Expected Outcome: The video plays without errors.

UCS-2: Voting on a Legislative Topic

- Actor: Website Visitor
- Precondition: The user has watched a video.
- Steps:

    UCS-2.1: User selects a vote option (e.g., Support/Oppose).

    UCS-2.2: The system records the vote in the local database.

    UCS-2.3: A confirmation message appears.

- Expected Outcome: The vote is stored, and a confirmation appears.

UCS-3: Submitting Feedback

- Actor: Website Visitor
- Precondition: The user has accessed the feedback form.
- Steps:

UCS-3.1: User enters comments into the feedback field.

UCS-3.2: User clicks submit.

UCS-3.3: The system saves the feedback.

UCS-3.4: A confirmation message appears.

- Expected Outcome: The feedback is recorded successfully.

UCS-4: ZIP Code Lookup for Location Filtering

- Actor: Website Visitor
- Precondition: The user has accessed the location search section.
- Steps:

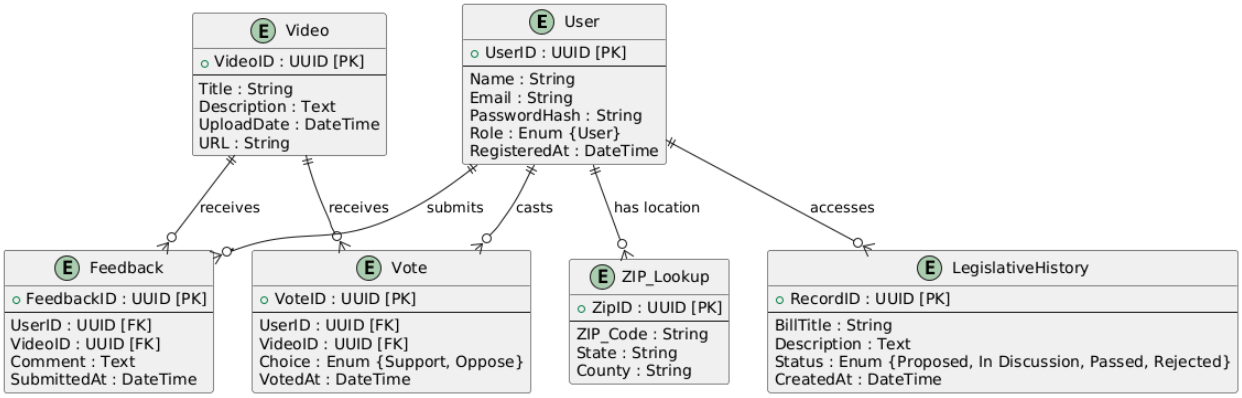
UCS-4.1: User enters a ZIP code.

UCS-4.2: The system fetches data from the API.

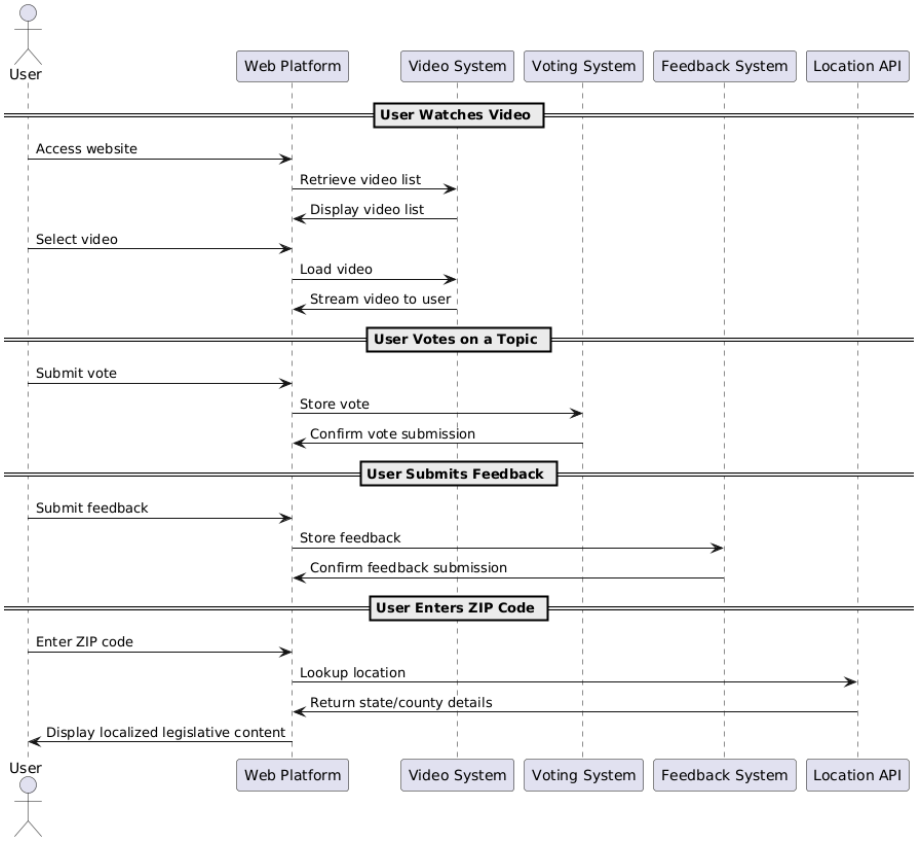
UCS-4.3: The correct state/county is displayed.

- Expected Outcome: The system correctly maps the ZIP code to a location.

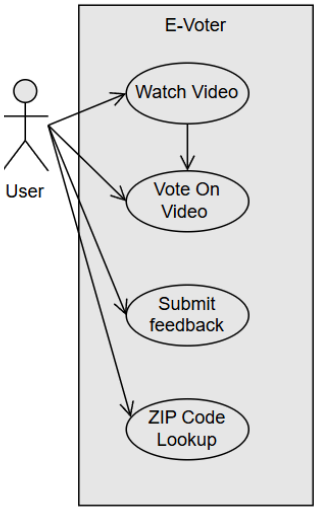
Entity Relationship Diagram



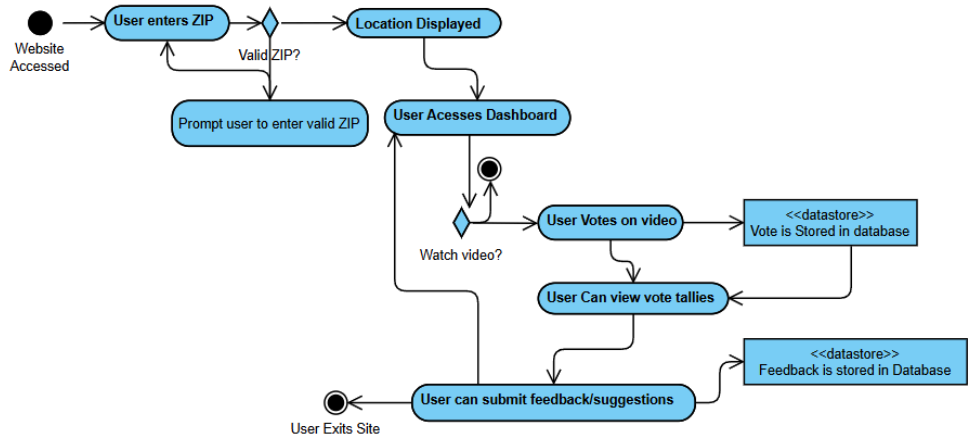
Sequence Diagram



Use Case Diagram



Activity Diagram



API Endpoint & Data Models

For our web app, we currently plan to use Zippopotam to provide simple location data. Information can be found here:

https://api.zippopotam.us

The search function uses information such as a zip code to give more detailed location information.

GET http://api.zippopotam.us/us/<zip code>

For example, entering: https://api.zippopotam.us/us/98027

returns: {"post code": "98027", "country": "United States", "country abbreviation": "US", "places": [{"place name": "Issaquah", "longitude": "-122.0107", "state": "Washington", "state abbreviation": "WA", "latitude": "47.4974"}]}

From this data we plan to extract the City and State information and display it on the page.

Test Plan

Objectives:

- Verify that core prototype functionalities work as expected.
- Ensure users can watch videos, vote, submit feedback, and perform ZIP code lookups.
- Identify potential usability and functionality issues before expanding the system.

Test Environment:

- Browser-based testing on Chrome, Firefox, and Edge.
- Desktop and mobile responsiveness tests.

Testing Methods:

- Functional Testing – Ensuring all core features operate as expected.
- Usability Testing – Checking if users can navigate and interact with the system easily.

Participants:

- One of each of the group members' family members if available, totalling at least 4.
- At least one should attempt to access the site from a handheld/mobile device

Tests:

Task 1: Watching a Legislative Video (T-1)

- Scenario: A user visits the website to watch a pre-recorded legislative debate.
- Steps:
    1. Navigate to the homepage.
    2. Select a video from the list.
    3. Play the video and interact with playback controls.
- Success Criteria: The video loads and plays without errors.

Task 2: Voting on a Legislative Topic (T-2)

- Scenario: A user wants to express their opinion on a legislative matter.
- Steps:
    1. Watch a video until the voting prompt appears.
    2. Select a voting option (Support/Oppose).
    3. Submit the vote and receive confirmation.
- Success Criteria: Vote is recorded, and confirmation message appears.

Task 3: Submitting Feedback (T-3)

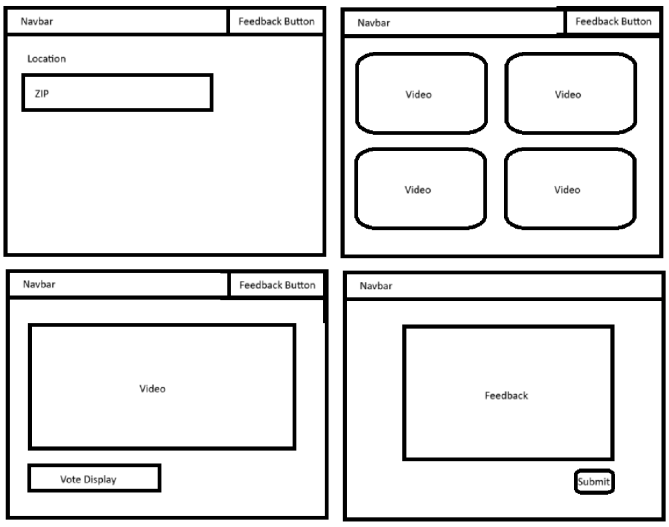
- Scenario: A user wants to provide feedback about their experience using the platform.
- Steps:
    1. Click the "Submit Feedback" button.
    2. Enter comments in the feedback form.
    3. Submit the form and receive confirmation.

- Success Criteria: Feedback is recorded and confirmation message appears.

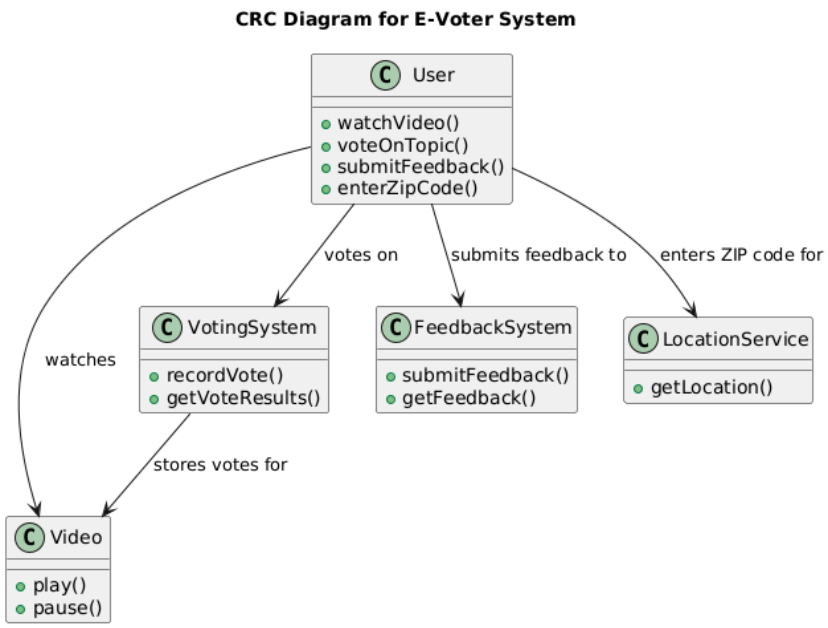
Task 4: ZIP Code Lookup for Location Filtering (T-4)

- Scenario: A user wants to check how location-based filtering works.
- Steps:
    1. Navigate to the location section.
    2. Enter a ZIP code.
    3. View the displayed state or county information.
- Success Criteria: The system correctly maps the ZIP code to a location.

Interface Wireframe



Class Responsibility Collaborator



Use Case Scenario Cards (UCSC)

UCSC-1: User is given Video and Prompted

Actor: User

Preconditions: User starts the program.

Basic Flow:

-The program provides a video for the user.

-The user watches through the video.

-The user is prompted to cast a vote and provide feedback.

Postconditions: User now knows what to vote for and is able to vote for it.

UCSC-2: User enters Vote

Actor: User

Preconditions: User was prompted to enter their vote.

Basic Flow:

-User provides their vote into an input system.

-System registers input, adds it to data for the video in question.

Postconditions: Vote is received and added to the video.

UCSC-3: Feedback from User is Received

Actor: System

Preconditions: User is prompted to provide feedback.

Basic Flow:

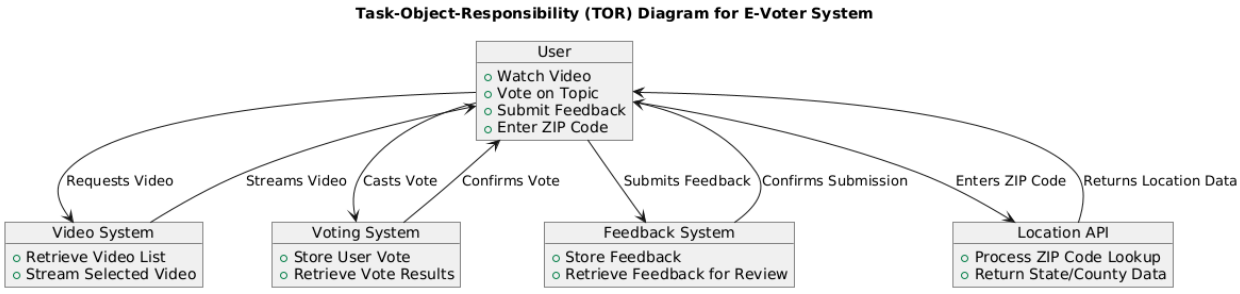
-User inputs feedback.

-Feedback is processed and stored by the system with proper context on what they voted on and their vote.

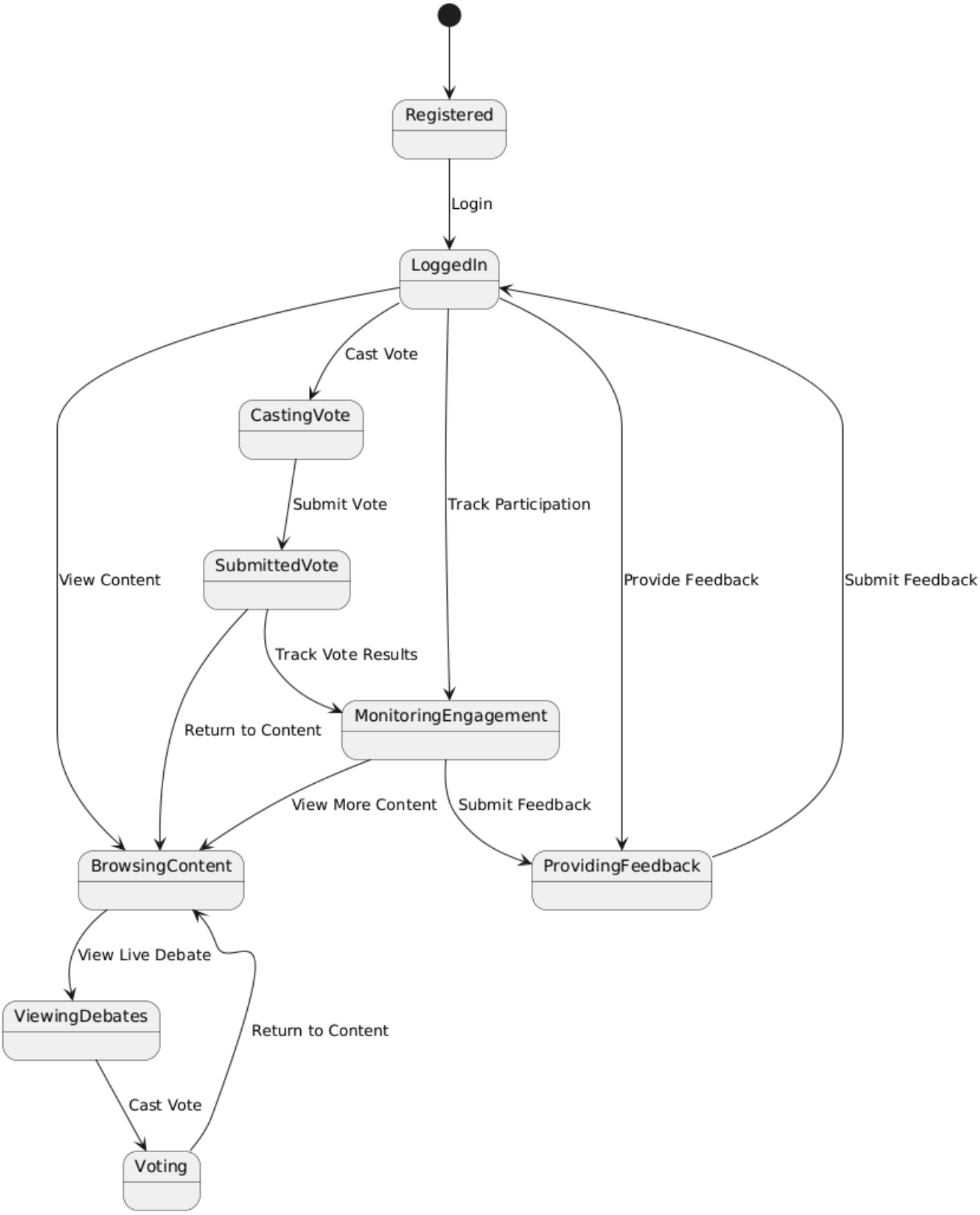
-Developers can review the feedback and improve what the users vote on or the system.

Postconditions: System is improved.

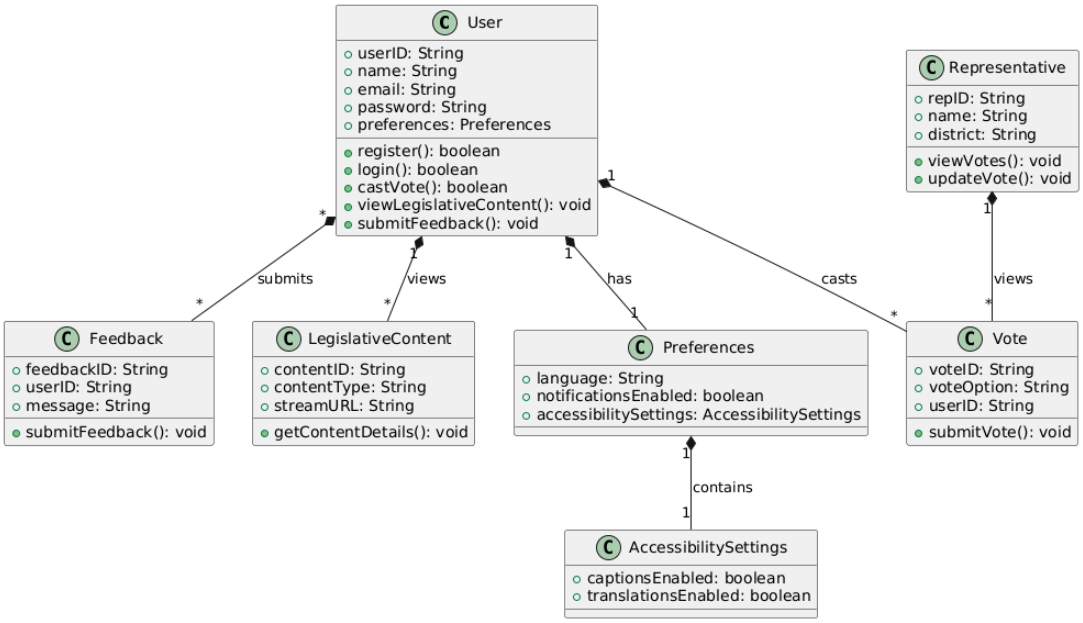
Task-Object-Responsibility (TOR) Diagram



State Diagram



Class Diagram



Preliminary Interface Diagram