

## **OOPs Final Project**

### **Civic Pulse - A Citizen Engagement & Participatory Governance Portal**

Presented to: Dr. Kiran DC

School of Computational and Data Sciences,  
Vidyashilp University, Bengaluru, India  
Submission Date - 05/05/2025

Written By: Ayush Kumar (2023UG000116)  
Pruthviraj Shinde (2023UG000103)

## Table Of Contents

Page No.

1. Introduction.....	3
2. Problem Statement.....	4
• 2.1 Key Issues Include	
- Delayed Issue Reporting and Resolution	
- Limited Citizen Engagement	
- Lack of Transparency	
- Inefficient Resource Allocation	
3. Objective.....	6
4. Operations Requirements.....	
• 4.1 Registration	
• 4.2 Issue Reporting	
• 4.3 Proposal Submission	
• 4.4 Voting on Issue / Proposal	
• 4.5 Commenting on Issue / Proposal	
• 4.6 Review and Prioritization of Issue	
• 4.7 View Dashboard	
• 4.8 Report Generation	
5. Use Case Diagram.....	9
• 5.1 Overview	
• 5.2 Key Operations and Relationships	
• 5.3 Relationship Explained	
• 5.4 Use Cases	
6. Class Diagram.....	18
• 6.1 Types of Arrow with Meaning	
• 6.2 Overview of the Class Diagram	
7. Sequence Diagram.....	24
• 7.1 Registration Seq. Diagram	
• 7.2 Issue Reporting Seq. Diagram	
• 7.3 Proposal Submission Seq. Diagram	
• 7.4 Vote on Issue / Proposal Seq. Diagram	
• 7.5 Review of Issue / Proposal Seq. Diagram	
• 7.6 Prioritization Seq. Diagram	
8. Code.....	32
9. Outputs.....	60
10. Conclusion.....	89
11. References.....	90

## ACKNOWLEDGEMENT

We would like to express our profound gratitude to the Vice Chancellor of Vidyashilp University, **Dr. P. G. Babu**, and the Registrar, **Mr. Ramakrishnan**, for fostering an environment of academic excellence. We extend our heartfelt thanks to the Program Chair, **Dr. Smitha Rao**, School of Computational and Data Sciences, Vidyashilp University, Bengaluru, for her unwavering support and encouragement.

We are deeply indebted to **Dr. Kiran DC**, our esteemed professor for the course **Object-Oriented Programming**, for his exceptional guidance, insightful feedback, and continuous motivation throughout this project. His expertise and enthusiasm for the subject have been a source of inspiration and a cornerstone for the successful completion of our work.

Finally, we extend our sincere gratitude to all the aforementioned dignitaries and everyone who has directly or indirectly contributed to the successful completion of our project.

## **1. Introduction**

In many urban areas, particularly in developing countries like India, citizens frequently encounter significant challenges in reporting civic issues such as potholes, waterlogging, inadequate sanitation, or erratic power supply, and in effectively engaging with local governance bodies. These difficulties stem from inefficient and outdated communication channels, where traditional methods like manual complaint submissions, phone calls to helplines, or sporadic public meetings dominate. Such approaches are often time-consuming, prone to delays, and lack transparency, leaving citizens frustrated and governance bodies overwhelmed. The rapid pace of urbanization, coupled with increasing population density, intensifies these problems, placing immense pressure on already strained infrastructure. This is especially critical during monsoon seasons, when issues like potholes and waterlogging transform into urgent hazards, posing risks to public safety and mobility. The absence of a streamlined, accessible platform further complicates timely identification and resolution of these concerns, often resulting in prolonged neglect of critical civic needs.

Beyond reporting issues, citizen participation in proposing innovative solutions or voting on community initiatives remains severely limited. This lack of engagement undermines the principles of participatory governance, where community input is essential for addressing local challenges effectively. In many cases, governance bodies operate in silos, with little mechanism to harness collective wisdom or prioritize projects based on public consensus. This disconnect is particularly evident in densely populated urban centers, where diverse needs and opinions go unheard, leading to inequitable resource allocation and unaddressed grievances. Moreover, the digital divide—where a significant portion of the population lacks access to technology or the skills to use it—exacerbates these issues, leaving marginalized communities further disenfranchised. The absence of a robust, inclusive system to bridge this gap not only hinders civic accountability but also stifles the potential for collaborative problem-solving, making the need for an efficient, transparent, and participatory platform more pressing than ever.

## **2. Problem Statement**

The absence of a centralized, accessible, and interactive platform severely impedes effective communication between citizens and local authorities, creating a significant barrier to efficient civic governance. This gap in infrastructure and engagement mechanisms leads to a cascade of challenges that undermine the quality of urban life and the responsiveness of governance systems. The lack of a unified system not only complicates the reporting and resolution process but also erodes trust between communities and administrative bodies, particularly in rapidly growing urban environments where demands are ever-increasing.

### **Key Issues Include:**

- **Delayed Issue Reporting and Resolution:** Citizens face considerable difficulties in reporting civic problems such as potholes, waterlogging, or sanitation issues in a timely manner due to reliance on fragmented and outdated channels like phone calls, physical complaint boxes, or infrequent public meetings. These methods often result in delays, as issues may go unnoticed or unreported for days or weeks. On the authorities' side, the absence of a systematic approach to collect, categorize, and prioritize these complaints leads to prolonged resolution times. This inefficiency is particularly acute during emergencies, such as monsoon seasons, where urgent hazards like flooded roads or collapsed infrastructure require immediate attention but are often addressed only after significant escalation or public outcry.
- **Limited Citizen Engagement:** Opportunities for citizens to actively participate in governance are severely restricted, with minimal avenues to propose innovative solutions, provide feedback through comments, or vote on community initiatives. This lack of involvement fosters a sense of disengagement, as citizens feel their voices are unheard and their contributions undervalued. In many urban areas, especially in developing regions, the absence of structured participatory mechanisms prevents the harnessing of local knowledge and ideas, which could otherwise lead to more tailored and effective solutions. This disconnection is further compounded by the digital divide, where a significant portion of the population lacks access to technology or the skills to engage, leaving marginalized groups particularly excluded from decision-making processes.

- **Lack of Transparency:** There is no robust mechanism in place for citizens to track the status of reported issues or for authorities to share regular progress updates, creating a veil of opacity around civic management. Without real-time visibility into whether a reported pothole has been scheduled for repair or a proposal is under consideration, citizens are left in the dark, fueling distrust and frustration. Similarly, authorities struggle to communicate their efforts or constraints, which hinders accountability and public support. This lack of transparency often results in repeated complaints about the same issues, as citizens have no way to verify resolution, perpetuating a cycle of inefficiency and dissatisfaction.
- **Inefficient Resource Allocation:** Without a structured system to assess and prioritize civic issues based on factors such as urgency, location, or community impact, authorities often allocate resources haphazardly. This misallocation can lead to wasted efforts, where critical problems in high-traffic areas are overlooked in favor of less pressing concerns in remote locations, or where funds are expended on incomplete solutions due to poor planning. The absence of data-driven prioritization—such as identifying "pothole during rain" as a high-priority issue—exacerbates resource wastage, straining municipal budgets and delaying infrastructure improvements. This inefficiency not only hampers service delivery but also erodes public confidence in governance effectiveness.

### **3. Objective**

The "Civic Pulse" portal is designed to address these challenges by offering a digital platform that fosters effective civic engagement and governance. It aims to enable seamless reporting of civic issues by citizens, allowing them to quickly and easily submit concerns such as potholes or waterlogging. The platform promotes active participation by providing tools for citizens to propose solutions, add comments, and vote on initiatives, thereby enhancing community involvement. It ensures transparent tracking and resolution processes, enabling authorities to manage and update issue statuses efficiently. Additionally, the portal facilitates the generation of comprehensive reports, empowering better governance and informed decision-making to optimize resource allocation and address urban challenges effectively.

The "Civic Pulse" portal aims to bridge this gap by providing a digital platform that facilitates:

- Seamless reporting of civic issues by citizens.
- Active participation through proposals, comments, and voting.
- Transparent tracking and resolution processes managed by authorities.
- Generation of reports for better governance and decision-making.

## **4. Operations Required**

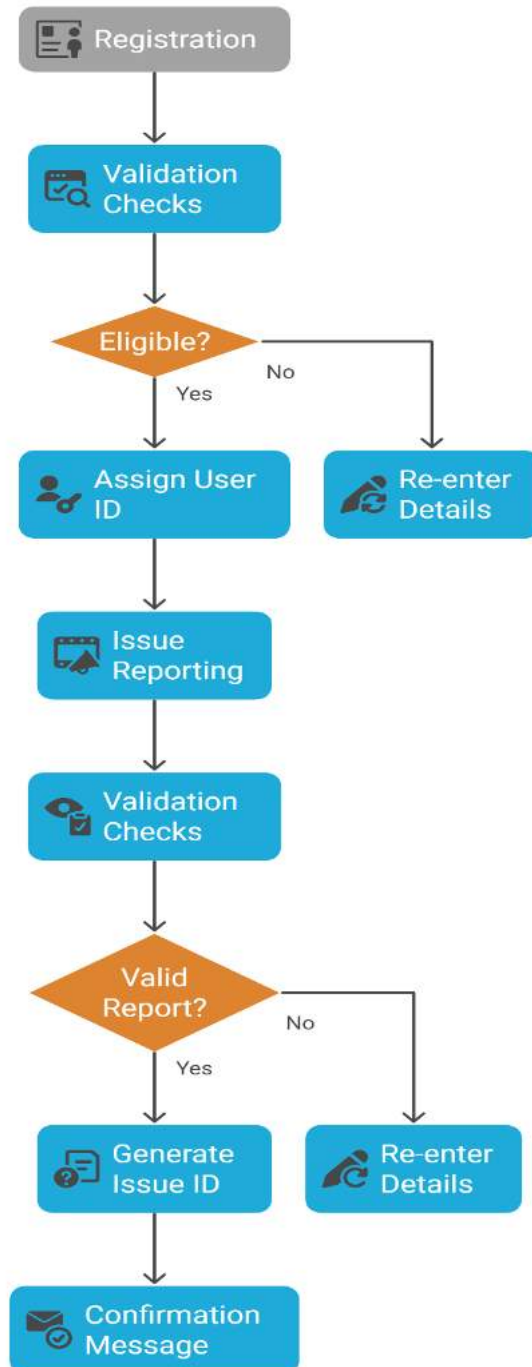
### **1.Registration**

- Enable citizens to create an account on the "Civic Pulse" platform by providing essential personal details such as name, gender, age, email, aadhar number (12 digits), mobile number, and address (including country, state, district, pin code, and locality).
- Include validation checks to ensure eligibility (e.g., age  $\geq 18$ ) and data integrity (e.g., valid aadhar format).
- Assign a unique user ID upon successful registration and provide a confirmation message.

### **2. Issue Reporting**

- Allow citizens to report civic problems (e.g., potholes, waterlogging) by submitting a description and location details.
- Incorporate validation to ensure the description is non-empty and the location is valid.
- Generate a unique issue ID and notify the citizen with a confirmation, enabling tracking of the reported issue.

## Civic Pulse Platform Operations

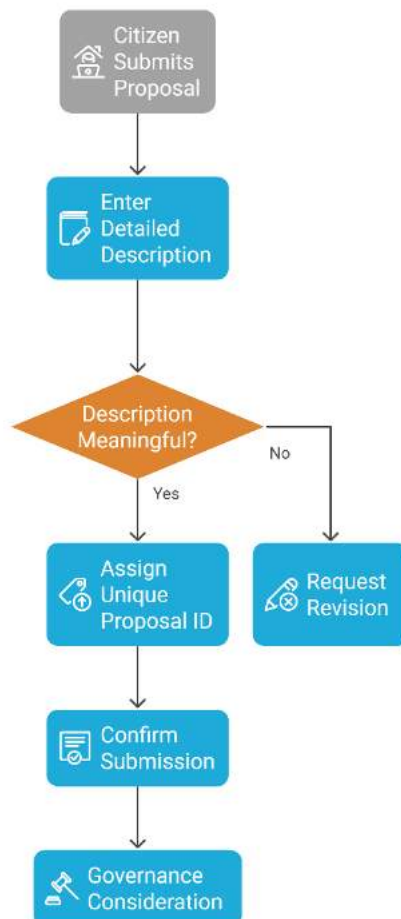




### 3.Proposal Submission

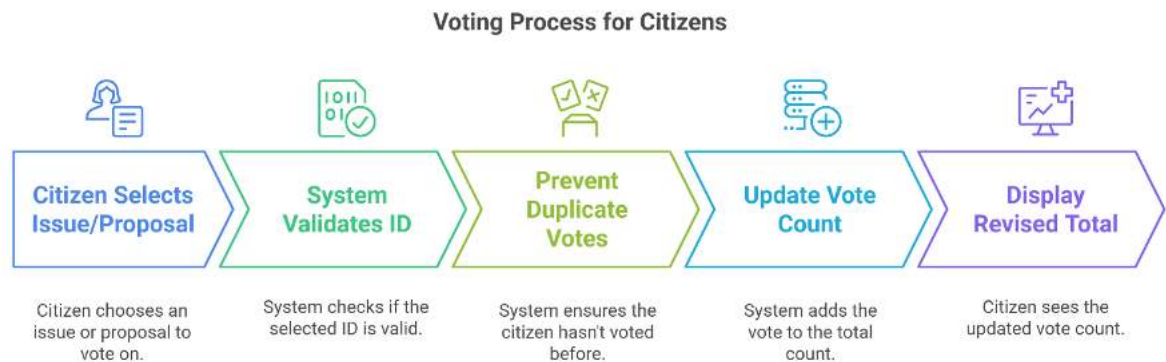
- Provide a mechanism for citizens to submit proposals for community improvements (e.g., building a park) by entering a detailed description.
- Include validation to ensure the description is meaningful and not empty.
- Assign a unique proposal ID and confirm submission, allowing citizens to contribute ideas for governance consideration.

Proposal Submission Process



## 4.Voting on Issue / Proposal

- Enable citizens to vote on reported issues or submitted proposals by selecting a valid issue or proposal ID.
- Implement checks to prevent duplicate votes and validate the existence of the selected ID.
- Update the vote count and display the revised total to the citizen, fostering democratic participation.



## 5.Commenting on Issue / Proposal

- Allow citizens to add comments on existing issues or proposals by providing a valid ID and text input.
- Include validation to ensure the ID exists and the comment is non-empty.
- Record the comment with a timestamp and notify the citizen, encouraging community dialogue and feedback.

### Citizen Commenting Process

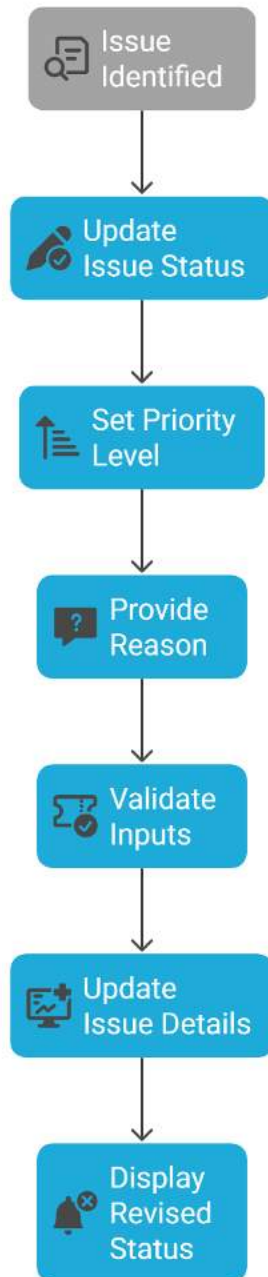


- 

## 6.Review and Prioritization of Issue

- Enable authorities to review issues by updating their status (e.g., "Under Review") using a valid issue ID.
- Allow authorities to prioritize issues by setting a priority level and providing a reason (e.g., "pothole during rain"), with validation for valid inputs.
- Update the issue details and display the revised status or priority to the authority, ensuring efficient issue management.

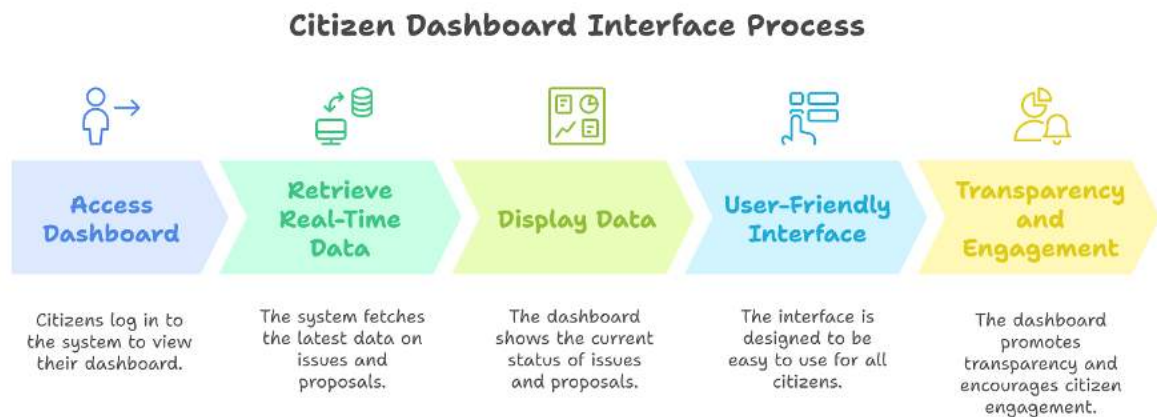
## Issue Review and Prioritization Process



•

## 7.View Dashboard

- Provide citizens with a dashboard view to monitor the status of issues, proposals, and their own activities.
- Retrieve and display real-time data (e.g., list of issues with statuses) from the system.
- Ensure the interface is user-friendly for CLI, offering a summary that supports transparency and engagement.

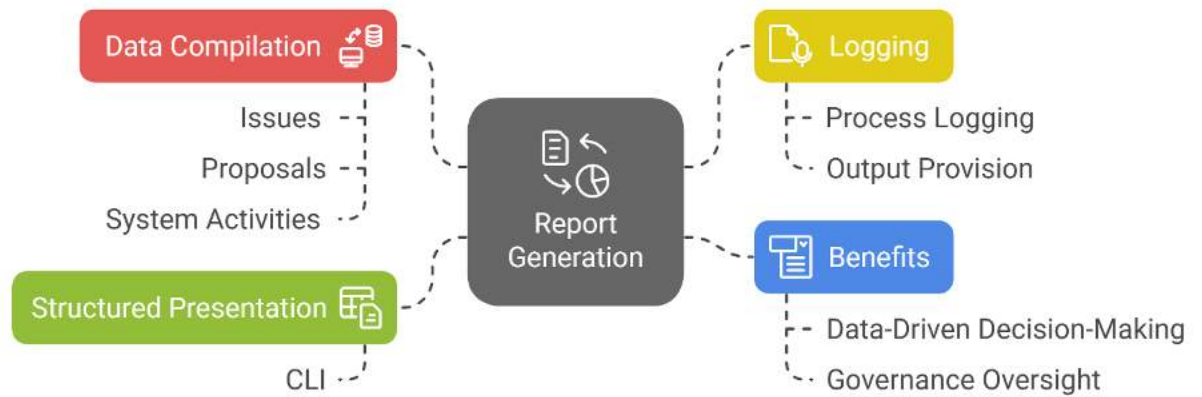


•

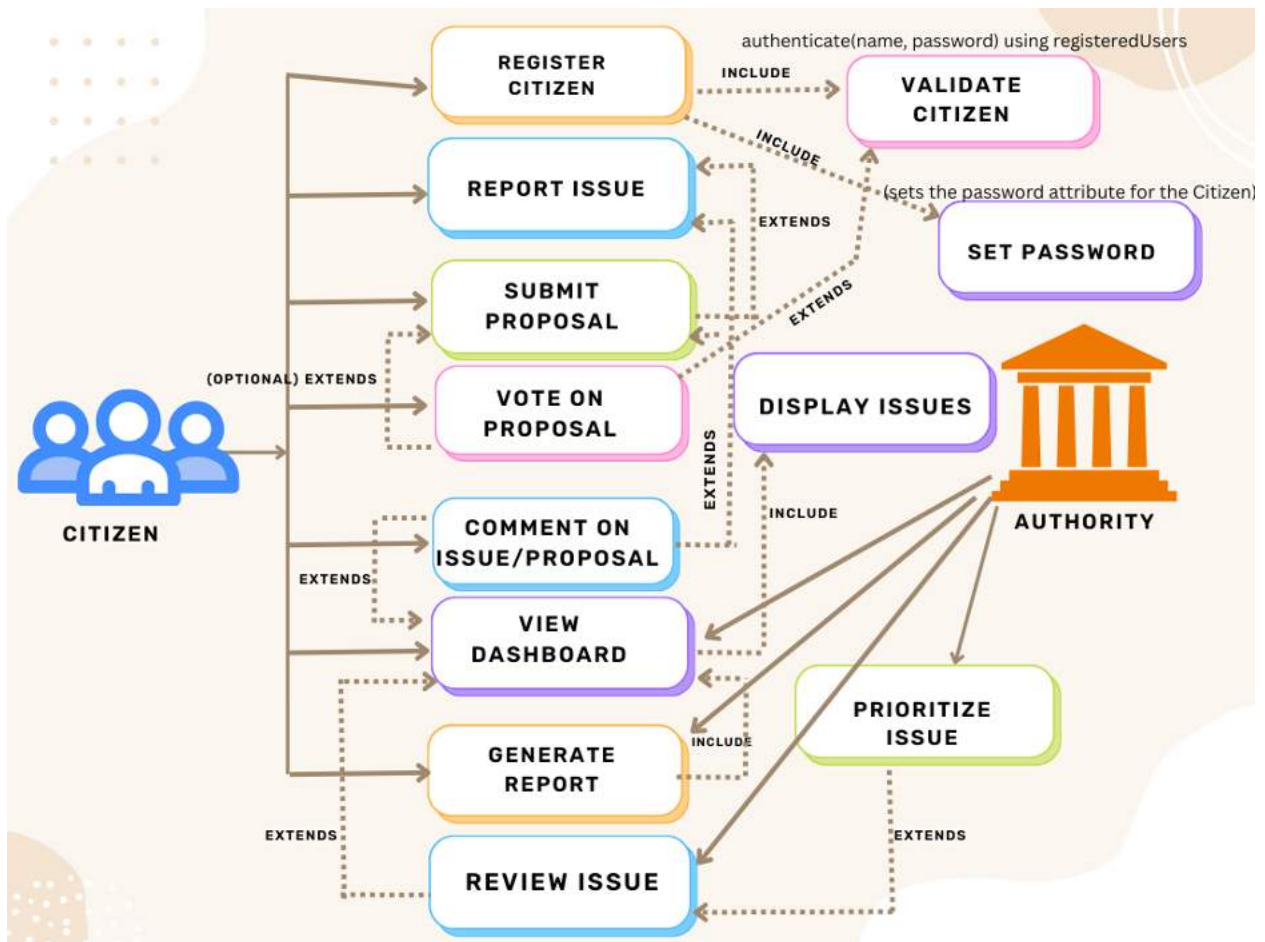
## 8.Report Generation

- Enable authorities to generate detailed reports on issues, proposals, and system activities (e.g., number of issues resolved).
- Compile data from the system and present it in a structured format via CLI.
- Log the report generation process and provide the output to support data-driven decision-making and governance oversight.

## Report Generation Process and Benefits



## 5. USE CASE DIAGRAM



## Overview

- **Actors:**
  - **Citizen:** The primary user who engages with the system.
  - **Authority:** Representing administrative or governmental entities managing the system.
- **Operations:** Listed as rectangular boxes, each representing a specific action or use case.

## Key Operations and Relationships

### 1. Register Citizen:

- **Description:** Allows a Citizen to sign up with personal details (e.g., name, age) and set a password for authentication.
- **Relationship:** Includes "Validate Citizen," indicating input validation (e.g., age  $\geq 18$ ), and includes "Set Password" to establish the password attribute, ensuring secure onboarding.
- **Purpose:** Onboards users into the system with verified credentials.

### 2.Report Issue:

- **Description:** Enables a Citizen to report civic problems (e.g., potholes).
- **Relationship:** Extends to other citizen actions, suggesting it's a core engagement activity.
- **Purpose:** Initiates issue tracking.

### 3.Submit Proposal:

- **Description:** Allows a Citizen to propose solutions (e.g., building a park).
- **Relationship:** Extends from citizen actions, indicating optional participation.
- **Purpose:** Encourages community input.

### 4.Vote on Proposal:

- **Description:** Lets a Citizen vote on submitted proposals or issues after authenticating with name and password using the registeredUsers list.
- **Relationship:** Includes "Display Issues," linking to authority oversight, and extends "Validate Citizen" to ensure authentication.
- **Purpose:** Supports democratic decision-making with secure participation.



#### 5.Comment on Issue/Proposal:

- **Description:** Allows a Citizen to add comments to issues or proposals.
- **Relationship:** Extends from citizen engagement, tied to displayed issues.
- **Purpose:** Facilitates discussion.

#### View Dashboard:

- **Description:** Enables a Citizen to see system status (e.g., issues, proposals).
- **Relationship:** Includes "Generate Report," linking to authority tasks.
- **Purpose:** Provides transparency.

#### 6.Generate Report:

- **Description:** Allows an Authority to create summary reports.
- **Relationship:** Extends from dashboard viewing, tied to review processes.
- **Purpose:** Supports administrative analysis.

#### 7.Review Issue:

- **Description:** Enables an Authority to update the status of an issue.
- **Relationship:** Extends from authority responsibilities, linked to prioritization.
- **Purpose:** Ensures issue resolution

#### 8.Prioritize Issue:

- **Description:** Allows an Authority to set issue priority (e.g., with a reason like "pothole during rain").
- **Relationship:** Extends from review, tied to displayed issues.
- **Purpose:** Addresses urgent civic needs.

## 9.Display Issues:

- **Description:** Shows current issues to both Citizen and Authority.
  - **Relationship:** Includes voting and commenting, serves as a central hub.
  - **Purpose:** Facilitates system-wide visibility.
- 

## Relationships Explained

- **Extends:** Dashed arrows (e.g., from Citizen to Report Issue) indicate that an operation is an extension of the actor's role, often optional or specialized.
- **Includes:** Solid arrows (e.g., from Register Citizen to Validate) show that an operation depends on or incorporates another process.
- **Links:** Lines connecting Citizen and Authority to Display Issues suggest shared access to system data.

## 1. Use Case: Register Citizen

- **Actors:**
  - **Citizen (User):** The individual registering to use the system.
- **Preconditions:**
  - The "Civic Pulse" system must be online and operational.
  - The citizen must provide all required fields in a valid format.
  - The citizen's age must be greater than or equal to 18.
  - The citizen must provide a valid aadhar card number (exactly 12 digits).
  - The citizen must not already be registered.
- **Main Flow:**
  - The Citizen logs into the "Civic Pulse" system via CLI (e.g., enters a temporary guest mode).
  - The Citizen selects the "Register Citizen" option.
  - The Citizen enters details: name, gender, age, email, aadharNo, mobileNo, address (country, state, district, pinCode, locality), and password.
  - The system validates the input (e.g., age  $\geq$  18, aadharNo as 12 digits) and throws `IllegalArgumentException` if invalid.

- If valid, the system sets the password, assigns a unique userId, and adds the Citizen to its registeredUsers list.
- The system displays a success message (e.g., "User 101 registered with name: Amit").
- **Postconditions:**
  - A new Citizen is added to the system with a unique userId and password.
  - The registration transaction is logged for record-keeping.
  - No changes are made to existing data.
  - If validation fails, an error message is displayed (e.g., "Age must be 18+"), and no registration occurs.

## 2. Use Case: Report Issue

- **Actors:**
  - **Citizen (User):** The individual reporting a civic issue.
- **Preconditions:**
  - The Citizen must be authenticated (logged in with a valid userId and password).
  - The "Civic Pulse" system must be online and operational.
  - The Citizen must provide a valid description and location.
- **Main Flow:**
  - The Citizen logs into the "Civic Pulse" system via CLI.
  - The Citizen selects the "Report Issue" option.
  - The Citizen enters a description and location.
  - The system validates the input (e.g., non-empty description, valid location) and throws IllegalArgumentException if invalid.
  - If valid, the system creates a new Issue and assigns a unique issuelid.
  - The system displays the result (e.g., "Issue 1 reported: Pothole in Mumbai").
- **Postconditions:**
  - A new Issue is created with the provided description and location.
  - The transaction is logged for record-keeping.
  - No changes are made to other data.
  - If validation fails, an error message is displayed (e.g., "Description cannot be empty"), and no issue is created.

## 3. Use Case: Submit Proposal

- **Actors:**
  - **Citizen (User):** The individual submitting a proposal.
- **Preconditions:**
  - The Citizen must be authenticated (logged in with a valid userId and password).
  - The "Civic Pulse" system must be online and operational.
  - The Citizen must provide a valid proposal description.
- **Main Flow:**

- The Citizen logs into the "Civic Pulse" system via CLI.
- The Citizen selects the "Submit Proposal" option.
- The Citizen enters a description.
- The system validates the input (e.g., non-empty description) and throws `IllegalArgumentException` if invalid.
- If valid, the system creates a new Proposal and assigns a unique proposalId.
- The system displays the result (e.g., "Proposal 1 submitted: Build a park").
- **Postconditions:**
  - A new Proposal is created with the provided description.
  - The transaction is logged for record-keeping.
  - No changes are made to other data.
  - If validation fails, an error message is displayed (e.g., "Description cannot be empty"), and no proposal is created.

#### 4. Use Case: Vote on Issue/Proposal

- **Actors:**
  - **Citizen (User):** The individual casting a vote.
- **Preconditions:**
  - The Citizen must be authenticated (logged in with a valid userId and password using the `authenticate(name, password)` method and `registeredUsers` list).
  - The "Civic Pulse" system must be online and operational.
  - The Citizen must provide a valid id for an existing Issue or Proposal.
  - The Citizen must not have voted on the same item previously (optional rule).
- **Main Flow:**
  - The Citizen logs into the "Civic Pulse" system via CLI.
  - The Citizen selects the "Vote on Proposal" option.
  - The Citizen enters an id (e.g., issue or proposal ID).
  - The system validates the input (e.g., existing id, no duplicate vote) and throws `IllegalArgumentException` if invalid.
  - If valid, the system increments the voteCount in the Issue or Proposal.
  - The system displays the result (e.g., "Voted on Issue 1, new count: 1").
- **Postconditions:**
  - The voteCount of the specified Issue or Proposal is incremented.
  - The transaction is logged for record-keeping.
  - No changes are made to other data.
  - If validation fails, an error message is displayed (e.g., "Invalid ID"), and no vote is recorded.

## 5. Use Case: Comment on Issue/Proposal

- **Actors:**
  - **Citizen (User):** The individual adding a comment.
- **Preconditions:**
  - The Citizen must be authenticated (logged in with a valid userId and password).
  - The "Civic Pulse" system must be online and operational.
  - The Citizen must provide a valid id for an existing Issue or Proposal and a non-empty comment text.
- **Main Flow:**
  - The Citizen logs into the "Civic Pulse" system via CLI.
  - The Citizen selects the "Comment on Issue/Proposal" option.
  - The Citizen enters an id and text.
  - The system validates the input (e.g., existing id, non-empty text) and throws `IllegalArgumentException` if invalid.
  - If valid, the system adds the comment to the Issue or Proposal.
  - The system displays the result (e.g., "Comment added to Issue 1: Urgent!").
- **Postconditions:**
  - A new comment is added to the specified Issue or Proposal.
  - The transaction is logged for record-keeping.
  - No changes are made to other data.
  - If validation fails, an error message is displayed (e.g., "Invalid ID"), and no comment is added.

## 6. Use Case: View Dashboard

- **Actors:**
  - **Citizen (User):** The individual viewing system status.
- **Preconditions:**
  - The Citizen must be authenticated (logged in with a valid userId and password).
  - The "Civic Pulse" system must be online and operational.
- **Main Flow:**
  - The Citizen logs into the "Civic Pulse" system via CLI.
  - The Citizen selects the "View Dashboard" option.
  - The system retrieves the list of issues and proposals from its data store.
  - The system displays the dashboard (e.g., "Issues: - ID: 1, Description: Pothole").
  - The system includes a report generation option (linked to Authority).
- **Postconditions:**
  - The dashboard with current issues and proposals is displayed to the Citizen.

- The transaction is logged for record-keeping.
- No changes are made to the data.
- The system remains ready for further interactions.

## 7. Use Case: Review Issue

- **Actors:**
  - **Authority:** The administrative entity reviewing an issue.
- **Preconditions:**
  - The Authority must be authenticated with administrative credentials.
  - The "Civic Pulse" system must be online and operational.
  - The Authority must provide a valid issuelid for an existing issue.
- **Main Flow:**
  - The Authority logs into the "Civic Pulse" system via CLI.
  - The Authority selects the "Review Issue" option.
  - The Authority enters an issuelid and a new status.
  - The system validates the input (e.g., existing issuelid) and throws `IllegalArgumentException` if invalid.
  - If valid, the system updates the status in the Issue.
  - The system displays the result (e.g., "Issue 1 status updated to Under Review").
- **Postconditions:**
  - The status of the specified Issue is updated.
  - The transaction is logged for record-keeping.
  - No changes are made to other data.
  - If validation fails, an error message is displayed (e.g., "Invalid Issue ID"), and no update occurs.

## 8. Use Case: Prioritize Issue

- **Actors:**
  - **Authority:** The administrative entity prioritizing an issue.
- **Preconditions:**
  - The Authority must be authenticated with administrative credentials.
  - The "Civic Pulse" system must be online and operational.
  - The Authority must provide a valid issuelid for an existing issue and a valid reason (e.g., "pothole during rain").
- **Main Flow:**
  - The Authority logs into the "Civic Pulse" system via CLI.
  - The Authority selects the "Prioritize Issue" option.
  - The Authority enters an issuelid, priority, and reason.

- The system validates the input (e.g., existing issueId, valid reason) and throws IllegalArgumentException if invalid.
- If valid, the system updates the priority and reason in the Issue.
- The system displays the result (e.g., "Issue 1 priority set to High, reason: pothole during rain").
- **Postconditions:**
  - The priority and reason of the specified Issue are updated.
  - The transaction is logged for record-keeping.
  - No changes are made to other data.
  - If validation fails, an error message is displayed (e.g., "Invalid reason"), and no update occurs.

## 9. Use Case: Generate Report

- **Actors:**
  - **Authority:** The administrative entity generating a report.
- **Preconditions:**
  - The Authority must be authenticated with administrative credentials.
  - The "Civic Pulse" system must be online and operational.
- **Main Flow:**
  - The Authority logs into the "Civic Pulse" system via CLI.
  - The Authority selects the "Generate Report" option.
  - The system retrieves data on issues, proposals, and transactions.
  - The system compiles and displays the report (e.g., "Report: 1 issues, 1 proposals as of 04/11/2025").
- **Postconditions:**
  - A report is generated and displayed to the Authority.
  - The transaction is logged for record-keeping.
  - No changes are made to the data.
  - The system remains ready for further interactions.





## 2. Dashed Line with a Hollow Arrowhead (..|>):

- **Meaning:** Represents **realization** or **implementation**. It indicates that the class at the tail of the arrow implements the interface or abstract class at the head. The class "implements" the contract defined by the interface.
- **Example in Diagram:**
  - CivicEngagement ..|> Engageable
  - CivicEngagement ..|> Trackable
- **Explanation:** CivicEngagement realizes the Engageable interface (implementing addComment) and the Trackable interface (implementing updateStatus). This enforces a contract that Issue and Proposal (as subclasses) must follow, ensuring consistent behavior.

## 3. Solid Line with a Diamond (o-->):

- **Meaning:** Represents **composition**, a strong form of aggregation where the container class (at the diamond end) owns the contained class, and the lifecycle of the contained object is dependent on the container. If the container is destroyed, the contained object is too.
- **Example in Diagram:**
  - User o--> Address
- **Explanation:** User has a Address object, indicating that each User instance owns an Address instance. If a User is deleted, its associated Address is also invalidated, reflecting a strong ownership relationship.

## 4. Solid Line with No Arrowhead (or with Multiplicity)--):

- **Meaning:** Represents **association**, a general relationship between two classes where one class uses or interacts with another. Multiplicity (e.g., many) indicates how many instances are involved (e.g., 0 or more, 1 or more).
- **Example in Diagram:**
  - CivicPulse o--> "many" User
  - CivicPulse o--> "many" Issue
  - CivicPulse o--> "many" Proposal
  - Issue o--> "many" Comment
  - Proposal o--> "many" Comment
- **Explanation:** CivicPulse is associated with multiple User, Issue, and Proposal instances, managing them as lists. Similarly, Issue and Proposal are associated with multiple Comment instances, indicating a one-to-many relationship. The diamond is optional here but emphasizes aggregation (a weaker form than composition).

#### 5. Dashed Line with No Arrowhead (or with Dependency):

- **Meaning:** Represents **dependency**, a weaker relationship where one class uses another but does not have a permanent association. Changes in the depended-upon class might affect the depending class.
  - **Example in Diagram:** Not explicitly used, but implied in method parameters (e.g., Citizen depends on Issue in reportIssue).
  - **Explanation:** This could be added if a method temporarily uses another class (e.g., Citizen passing data to CivicPulse), but it's typically implicit in method signatures in this diagram.
- 

### Summary of Arrow Usage in "Civic Pulse"

- **Inheritance (<|--)**: Establishes the "is-a" hierarchy (e.g., Citizen is a User), enabling code reuse and polymorphism.
- **Realization (..|>)**: Ensures CivicEngagement adheres to Engageable and Trackable contracts, promoting interface-based design.
- **Composition (o-->)**: Reflects strong ownership (e.g., User owns Address), ensuring data integrity.
- **Association (-->)**: Models the "has-a" or "manages-a" relationship (e.g., CivicPulse manages many Issue objects), supporting system coordination.

These arrows collectively define a robust structure where abstract classes and inheritance provide a foundation, while associations and composition manage relationships, aligning with the CLI-based "Civic Pulse" system's needs.

## Overview of the Class Diagram

The Class Diagram represents the structure of the "Civic Pulse" system, a platform designed to facilitate civic engagement through user registration, issue reporting, proposal submission, voting, commenting, and issue review/prioritization. It includes classes for users (e.g., Citizen, Authority), engagement entities (e.g., Issue, Proposal), and system management (CivicPulse), along with interfaces (Engageable, Trackable) and a nested Address class. The design leverages abstraction, inheritance, and composition to ensure reusability, extensibility, and maintainability.

## Use of Abstract Classes

Abstract classes are utilized to define a common blueprint for related classes, enforcing a structure that subclasses must adhere to while allowing shared behavior. In this diagram, three classes are abstract:

- **User (Abstract):**
  - **Attributes:** -userId: int, -name: String, -gender: String, -age: int, -email: String, -aadharNo: String, -mobileNo: String, -address: Address, -password: String.
  - **Methods:** +register(): void, +login(password: String): boolean.
  - **Justification:** The User class is abstract as it represents a generic user entity with core attributes (e.g., personal details, credentials including the newly added password) and behaviors (e.g., registration, login). The specific implementation varies between Citizen and Authority (e.g., different validation or access levels).

Abstraction ensures subclasses provide concrete implementations, promoting polymorphism and reducing redundancy. It also supports future user types (e.g., Moderator) without altering the core design.

- **Engagement (Abstract):**
  - **Attributes:** -id: int, -description: String.
  - **Methods:** +getDetails(): String.
  - **Justification:** Engagement is abstract to define a generic structure for all engagement-related entities (e.g., Issue, Proposal) sharing an identifier and description. The getDetails() method is abstract because the retrieval or formatting differs (e.g., location for Issue, vote count for Proposal). This abstraction provides a consistent interface while allowing specialized implementations, enhancing flexibility for future engagement types.
- **CivicEngagement (Abstract):**
  - **Attributes:** -comments: List<Comment>.
  - **Methods:** +addComment(text: String, citizenId: int): void, +updateStatus(status: String): void.
  - **Justification:** CivicEngagement is abstract to act as an intermediate layer in the multilevel inheritance hierarchy, extending Engagement and adding community-specific features (e.g., comments, status updates) applicable to both Issue and Proposal. The abstract methods ensure tailored implementations (e.g., different status validation). This supports extensibility, allowing new types (e.g., Petition) to inherit and customize these features.

## Use of Inheritance

Inheritance establishes a hierarchical relationship where subclasses inherit attributes and methods from superclasses, promoting code reuse and an "is-a" relationship. The diagram employs both single-level and multilevel inheritance:

- **Single-Level Inheritance:**
  - User <|-- Citizen and User <|-- Authority.
  - **Justification:** Both Citizen and Authority are specific types of User, inheriting common attributes (e.g., name, password) and methods (e.g., register, login). This reduces redundancy, with user-related data and authentication logic defined once in User. Citizen adds methods like reportIssue, submitProposal, and vote, tailored to civic participation, while Authority adds reviewIssue and prioritizeIssue for administrative roles. Inheritance ensures a clear role-based hierarchy built on the generic User foundation.
- **Multilevel Inheritance:**
  - Engagement <|-- CivicEngagement <|-- Issue and Engagement <|-- CivicEngagement <|-- Proposal.
  - **Justification:** This multilevel inheritance creates a three-tier hierarchy where Engagement provides a base for all engagement entities, CivicEngagement adds

community interaction features (e.g., comments, status), and Issue/Proposal implement domain-specific details (e.g., location for Issue, voteCount for Proposal). This structure mirrors the natural complexity progression, supports extensibility (e.g., adding Petition under CivicEngagement), and ensures efficient inheritance of shared behavior (e.g., addComment).

## Additional Design Elements

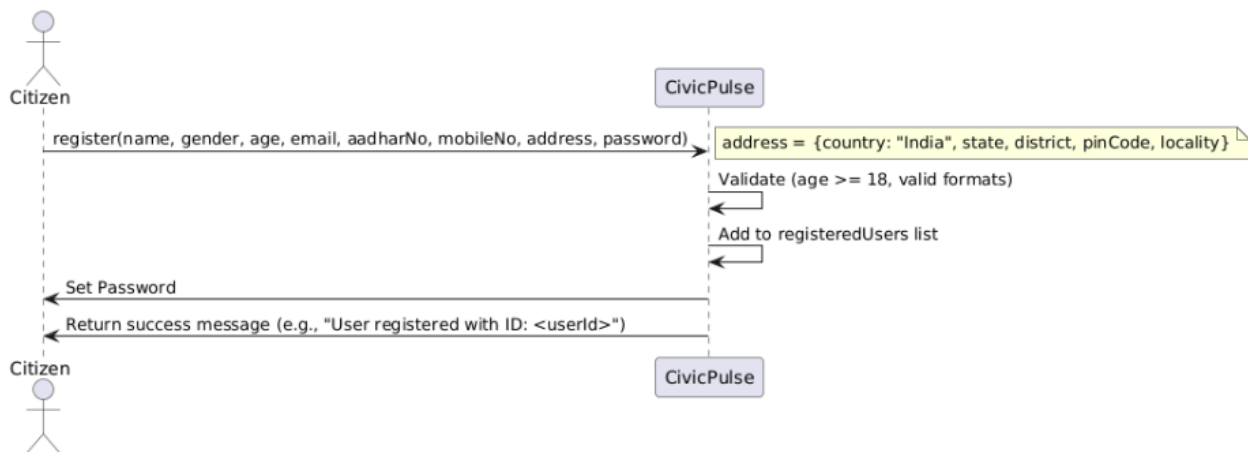
- **Interfaces (Engageable, Trackable):**
  - CivicEngagement realizes Engageable (addComment) and Trackable (updateStatus) to enforce these behaviors across Issue and Proposal, promoting a contract-based design for consistency and extensibility.
- **Composition (User o--> Address):**
  - User has an Address object, encapsulating details (e.g., country, state), which supports data organization and reusability.
- **Associations (CivicPulse to User, Issue, Proposal):**
  - CivicPulse manages collections of these entities, reflecting its role as the system coordinator.
- **Exception Handling:**
  - Methods include comments indicating potential exceptions (e.g., IllegalArgumentException), preparing for robust error handling in implementation.

## Why This Design Works

- **Abstraction:** By making User, Engagement, and CivicEngagement abstract, the design enforces a contract that subclasses must fulfill, reducing errors and supporting future expansion.
- **Inheritance:** The hierarchical structure (single-level for users, multilevel for engagements) minimizes code duplication and aligns with the system's domain (e.g., citizens/authorities as users, issues/proposals as engagements).
- **Flexibility:** Interfaces and multilevel inheritance allow the system to evolve (e.g., new user roles, engagement types) without major redesign.
- **Encapsulation:** Private attributes with public methods ensure data integrity and controlled access.
- **Maintainability:** The clear hierarchy and separation of concerns (e.g., user management vs. engagement handling) simplify updates and debugging.

## 7. SEQUENCE DIAGRAM

### 1. Registration Sequence Diagram



Code -

```

@startuml
actor Citizen
participant CivicPulse

```

```

Citizen -> CivicPulse: register(name, gender, age, email, aadharNo, mobileNo, address, password)
note right of CivicPulse: address = {country: "India", state, district, pinCode, locality}
CivicPulse -> CivicPulse: Validate (age >= 18, valid formats)
CivicPulse -> CivicPulse: Add to registeredUsers list
CivicPulse -> Citizen: Set Password
CivicPulse -> Citizen: Return success message (e.g., "User registered with ID: <userId>")

```

```

@enduml

```

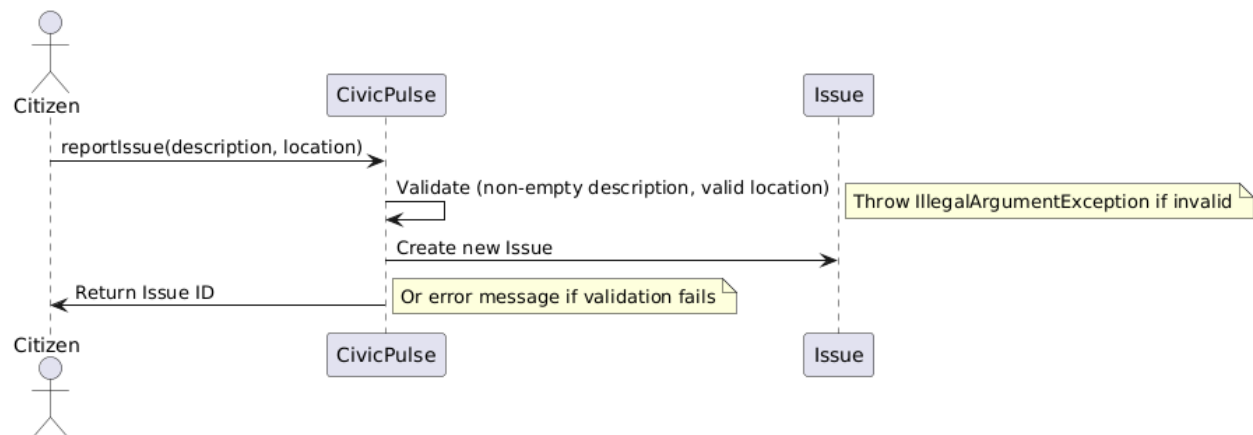
**Purpose:** Models the process of a Citizen registering with the "Civic Pulse" system, capturing personal details including a password and validating them.

**Flow:**

- Citizen sends register(name, gender, age, email, aadharNo, mobileNo, address, password) to CivicPulse.
- CivicPulse validates inputs (e.g., age  $\geq 18$ , aadharNo as 12 digits) and throws IllegalArgumentException if invalid.
- If valid, CivicPulse adds the user to its registeredUsers list, sets the password, and returns a success message (e.g., "User 101 registered").

**Explanation:** This diagram ensures new users are onboarded with verified data and a secure password, critical for system security and eligibility. The validation step reflects real-world constraints (e.g., age limit), and the addition of password setting enhances authentication integrity. The use of the registeredUsers list aligns with the updated class structure, making it robust for CLI feedback.

## 2. Issuing Report



Code -

```

@startuml
actor Citizen
participant CivicPulse
participant Issue
  
```

```

Citizen -> CivicPulse: reportIssue(description, location)
CivicPulse -> CivicPulse: Validate (non-empty description, valid location)
note right: Throw IllegalArgumentException if invalid
  
```

CivicPulse -> Issue: Create new Issue  
 CivicPulse -> Citizen: Return Issue ID  
 note right: Or error message if validation fails

@enduml

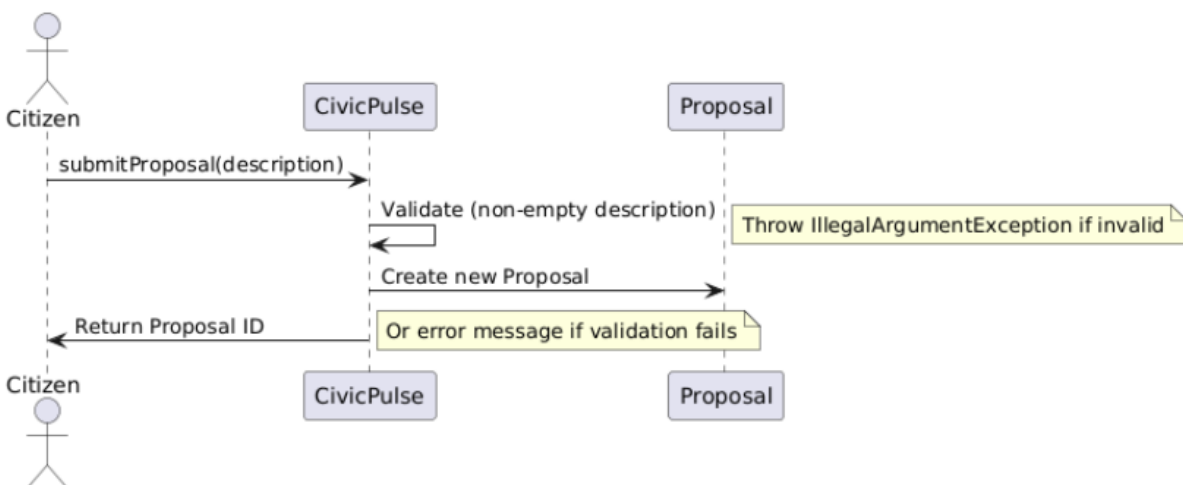
**Purpose:** Depicts a Citizen reporting a civic issue (e.g., pothole) to the "Civic Pulse" system.

**Flow:**

- Citizen sends reportIssue(description, location) to CivicPulse.
- CivicPulse validates (e.g., non-empty description, valid location) and throws IllegalArgumentException if invalid.
- If valid, CivicPulse creates a new Issue and returns its ID to Citizen.

**Explanation:** This sequence captures the initiation of civic engagement, ensuring issues are meaningful and locatable. Validation prevents incomplete reports, aligning with the system's goal of actionable data collection. The process remains unaffected by the updated Class Diagram changes (e.g., addition of password to User, registeredUsers list, and authenticate method in Citizen), as authentication is not a prerequisite for reporting in this context.

### 3. Submission Proposal



@startuml  
 actor Citizen



participant CivicPulse  
participant Proposal

Citizen -> CivicPulse: submitProposal(description)  
CivicPulse -> CivicPulse: Validate (non-empty description)  
note right: Throw IllegalArgumentException if invalid  
CivicPulse -> Proposal: Create new Proposal  
CivicPulse -> Citizen: Return Proposal ID  
note right: Or error message if validation fails

@enduml

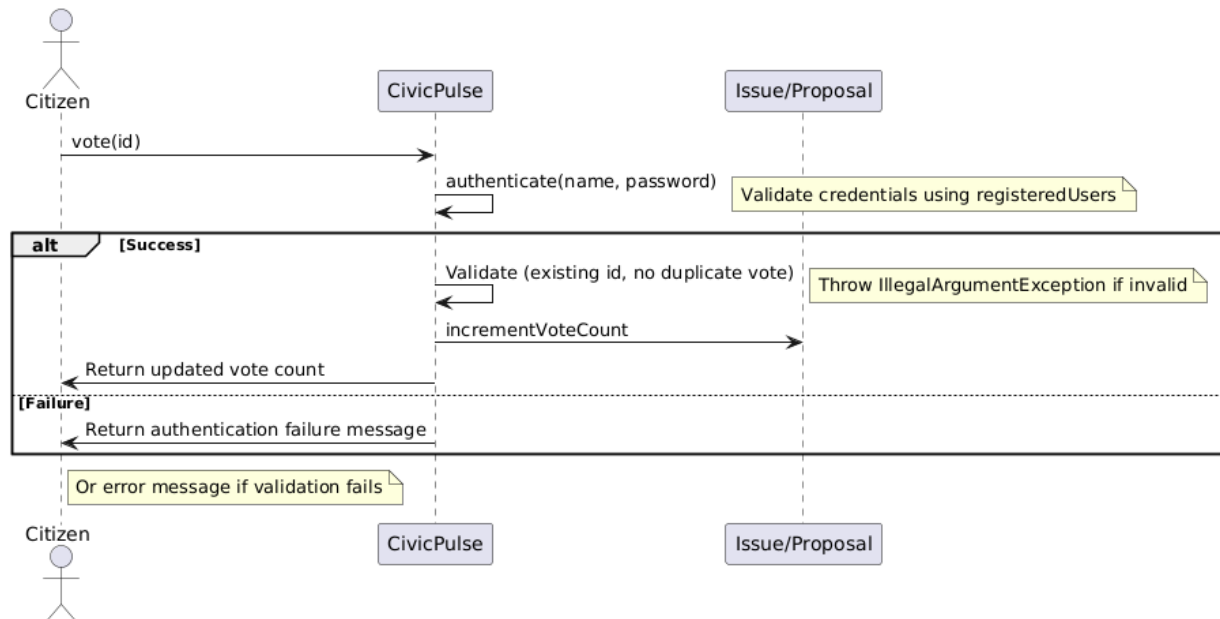
**Purpose:** Shows a Citizen voting on an existing Issue or Proposal to influence its priority or acceptance.

**Flow:**

- Citizen sends vote(id) to CivicPulse.
- CivicPulse validates (e.g., existing id, no duplicate vote) and throws IllegalArgumentException if invalid.
- If valid, CivicPulse increments the voteCount in Issue/Proposal and returns the updated count to Citizen.

**Explanation:** This sequence promotes community input, with validation preventing invalid or repeated votes. It's key to democratic decision-making in the system, reflected in CLI (Command Line Interface) output.

#### 4. Vote on Issue / Proposal



Code

```

@startuml
actor Citizen
participant CivicPulse
participant "Issue/Proposal"

Citizen -> CivicPulse: vote(id)
CivicPulse -> CivicPulse: authenticate(name, password)
note right: Validate credentials using registeredUsers
alt Success
    CivicPulse -> CivicPulse: Validate (existing id, no duplicate vote)
    note right: Throw IllegalArgumentException if invalid
    CivicPulse -> "Issue/Proposal": incrementVoteCount
    "Issue/Proposal" --> CivicPulse: 
    CivicPulse --> Citizen: Return updated vote count
else Failure
    CivicPulse --> Citizen: Return authentication failure message
end
note left of Citizen: Or error message if validation fails

@enduml
  
```

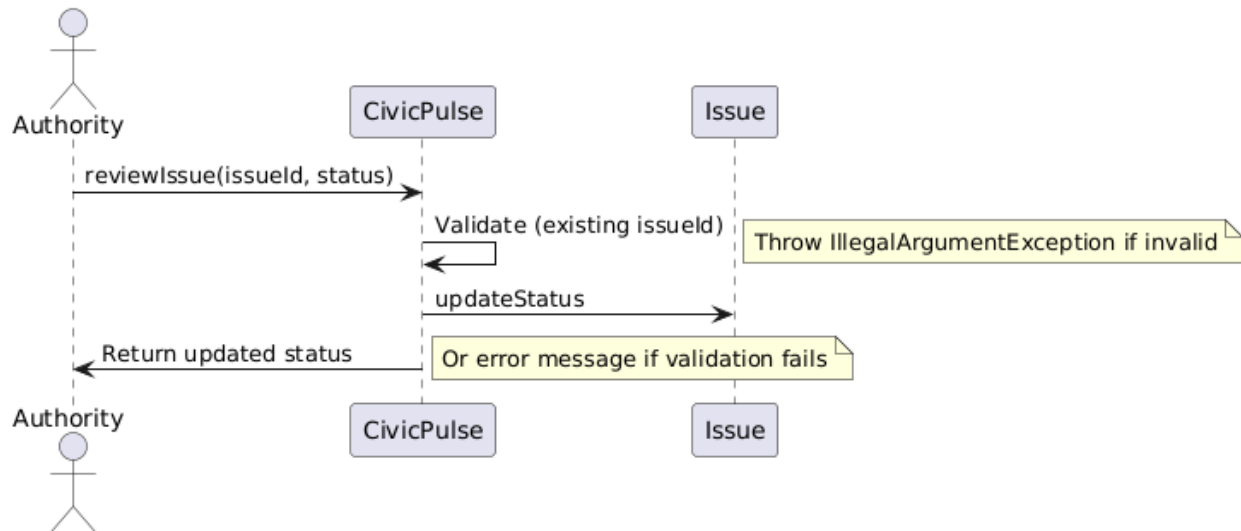
**Purpose:** Shows a Citizen voting on an existing Issue or Proposal to influence its priority or acceptance.

**Flow:**

- Citizen sends vote(id) to CivicPulse.
- CivicPulse authenticates the Citizen using authenticate(name, password) with the registeredUsers list and throws an exception if credentials are invalid.
- If authentication succeeds, CivicPulse validates (e.g., existing id, no duplicate vote) and throws IllegalArgumentException if invalid.
- If valid, CivicPulse increments the voteCount in Issue/Proposal and returns the updated count to Citizen.

**Explanation:** This sequence promotes community input, with validation and authentication preventing invalid, repeated, or unauthorized votes. The addition of authentication, leveraging the updated password attribute and authenticate method, enhances security, ensuring only registered and verified Citizens can participate. This is key to democratic decision-making in the system, reflected in CLI output, and aligns with the updated Class Diagram's focus on secure user management.

## 5. Review Issue Sequence Diagram



Code :

```

@startuml
actor Authority
participant CivicPulse
participant Issue
  
```

```

Authority -> CivicPulse: reviewIssue(issueId, status)
CivicPulse -> CivicPulse: Validate (existing issueId)
note right: Throw IllegalArgumentException if invalid
CivicPulse -> Issue: updateStatus
CivicPulse -> Authority: Return updated status
note right: Or error message if validation fails
  
```

```

@enduml
  
```

Purpose: Depicts an Authority reviewing and updating the status of an Issue.

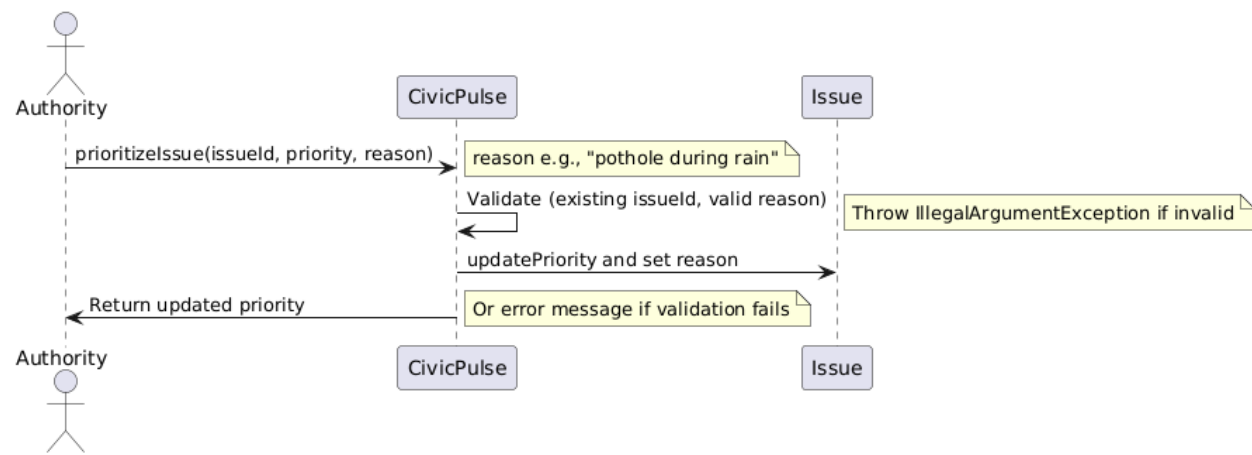
Flow:

- Authority sends reviewIssue(issueId, status) to CivicPulse.

- CivicPulse validates (e.g., existing issued) and throws `IllegalArgumentException` if invalid.
- If valid, CivicPulse updates the status in Issue and returns the updated status to Authority.

Explanation: This sequence enables authority oversight, ensuring issues are tracked and resolved. Validation guarantees the process targets valid issues, critical for administrative control. The updated Class Diagram (e.g., adding password to User, updating Citizen constructor, and adding registeredUsers and authenticate to Citizen) does not affect this flow, as no authentication requirement was specified for Authority actions, keeping the process focused on issue management.

## 6. Prioritizing Issue Sequence Diagram



Code :

```

@startuml
actor Authority
participant CivicPulse
participant Issue
  
```

```

Authority -> CivicPulse: prioritizeIssue(issuelid, priority, reason)
note right of CivicPulse: reason e.g., "pothole during rain"
CivicPulse -> CivicPulse: Validate (existing issuelid, valid reason)
note right of CivicPulse: Throw IllegalArgumentException if invalid
CivicPulse -> Issue: updatePriority and set reason
note right of Issue: Or error message if validation fails
Issue --> Authority: Return updated priority
  
```

note right: Or error message if validation fails

@enduml

**Purpose:** Illustrates an Authority prioritizing an Issue with a reason (e.g., "pothole during rain").

**Flow:**

- Authority sends `prioritizeIssue(issueId, priority, reason)` to `CivicPulse`.
- `CivicPulse` validates (e.g., existing `issueId`, valid reason) and throws `IllegalArgumentException` if invalid.
- If valid, `CivicPulse` updates the priority and reason in `Issue` and returns the updated priority to Authority.

**Explanation:** This sequence allows authorities to address urgent issues with context, enhancing decision-making. Validation ensures meaningful prioritization, aligning with the system's goal of effective issue management. The updated Class Diagram (e.g., adding password to `User`, updating `Citizen` constructor, and adding `registeredUsers` and `authenticate` to `Citizen`) does not affect this flow, as no authentication requirement was specified for Authority actions, keeping the focus on issue prioritization

## 8. Code -

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

// Custom Exception
class InvalidPriorityException extends Exception {
    public InvalidPriorityException(String message) {
        super(message);
    }
}
```

```
}
```

```
class Address {  
    private String state;  
    private String district;  
    private String locality;  
  
    public Address(String state, String district, String locality) {  
        this.state = state;  
        this.district = district;  
        this.locality = locality;  
    }  
  
    public String getState() { return state; }  
    public String getDistrict() { return district; }  
    public String getLocality() { return locality; }  
}
```

```
abstract class User {  
    protected String name;  
    private int age;  
    private String email;  
    private String aadharNo;  
    private String mobileNo;  
    private Address address;  
    private String password;  
  
    public User() {  
        this.name = "DefaultUser";  
        this.age = 0;  
        this.email = "";  
        this.aadharNo = "";  
        this.mobileNo = "";  
        this.address = new Address("", "", ""); // OBJECT: Address object created  
        this.password = "";  
    }  
  
    public User(String name, int age, String email, String aadharNo, String mobileNo, Address  
    address, String password) {  
        this.name = name;  
        this.age = age;  
        this.email = email;  
        this.aadharNo = aadharNo;
```

```
this.mobileNo = mobileNo;
this.address = address;
this.password = password;
}

public String getName() { return name; }
public int getAge() { return age; }
public String getEmail() { return email; }
public String getAadharNo() { return aadharNo; }
public String getMobileNo() { return mobileNo; }
public Address getAddress() { return address; }
public String getPassword() { return password; }
public abstract void authenticate();
}

class RegisteredUser extends User {
    public RegisteredUser(String name, int age, String email, String aadharNo, String mobileNo,
Address address, String password) {
        super(name, age, email, aadharNo, mobileNo, address, password);
    }

    @Override
    public void authenticate() {
        // Placeholder
    }
}

class Authority extends User {
    public Authority(String name, String password) {
        super(name, 0, "", "", "", new Address("", "", ""), password);
    }

    @Override
    public void authenticate() {
        // Placeholder
    }
}

interface Engageable {
    void addComment(String comment) throws IllegalArgumentException;
    void addComment(String comment, String author) throws IllegalArgumentException;
}
```



```
interface Trackable {  
    void updateStatus(String status) throws InvalidPriorityException;  
    String getStatus();  
}
```

```
interface Notifiable {  
    void notifyUser(String message);  
}
```

```
class Issue implements Engageable, Trackable {  
    private static int issueIdCounter = 1;  
    private static List<String> issues = new ArrayList<>();  
    private String description;  
    private Address location;  
    private static List<String> comments = new ArrayList<>();  
    private int voteCount = 0;  
    private String status = "Not Reviewed";  
    private String reportedBy;  
    private Date reportedTime;
```

```
    class CommentManager {  
        private void logComment(String comment) {  
            System.out.println("Logging comment internally for Issue " + issueIdCounter);  
        }  
    }  
}
```

```
    public Issue(String description, Address location, String reportedBy) {  
        this.description = description;  
        this.location = location;  
        this.reportedBy = reportedBy;  
        this.reportedTime = new Date();  
        issues.add(description + " in " + location.getState() + ", " + location.getDistrict() + ", " +  
location.getLocality());  
        new CommentManager().logComment("Initial comment setup");  
    }
```

```
    public static int getNextId() { return issueIdCounter++; }  
    public static List<String> getIssues() { return issues; }  
    public String getReportedBy() { return reportedBy; }  
    public Date getReportedTime() { return reportedTime; }  
    public Address getLocation() { return location; }  
    public void setDescription(String newDescription) {  
        this.description = newDescription;
```

```

        int index = issues.indexOf(getIssues().get(issues.indexOf(this.description + " in " +
location.getState() + ", " + location.getDistrict() + ", " + location.getLocality())));
        if (index >= 0) {
            issues.set(index, newDescription + " in " + location.getState() + ", " +
location.getDistrict() + ", " + location.getLocality());
        }
    }

    @Override
    public void addComment(String comment) throws IllegalArgumentException {
        if (comment == null || comment.trim().isEmpty()) {
            throw new IllegalArgumentException("Comment cannot be empty.");
        }
        SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
        String timestampedComment = comment + " on " + sdf.format(new Date());
        comments.add(timestampedComment);
        System.out.println("Comment added to Issue " + issueIdCounter + ": " + comment + " on "
+ sdf.format(new Date()));
    }

    @Override
    public void addComment(String comment, String author) throws IllegalArgumentException {
        if (comment == null || comment.trim().isEmpty()) {
            throw new IllegalArgumentException("Comment cannot be empty.");
        }
        SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
        String timestampedComment = comment + " by " + author + " on " + sdf.format(new
Date());
        comments.add(timestampedComment);
        System.out.println("Comment added to Issue " + issueIdCounter + ": " + comment + " on "
+ sdf.format(new Date()));
    }

    public void addVote(int votes) {
        this.voteCount += votes;
        updatePriority();
    }

    private void updatePriority() {
        if (voteCount <= 1) {
            status = "Not Reviewed";
        } else if (voteCount <= 5) {
            status = "Under Review";
        }
    }

```

```

    } else {
        status = "Reviewed";
    }
}

@Override
public void updateStatus(String status) throws InvalidPriorityException {
    if (!status.matches("Low|Medium|High")) {
        throw new InvalidPriorityException("Invalid priority value. Use Low, Medium, or High.");
    }
    this.status = status;
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    int displayId = issues.indexOf(description + " in " + location.getState() + ", " +
location.getDistrict() + ", " + location.getLocality()) + 1;
    System.out.println("Issue " + displayId + " updated: " + description + " - Priority: " +
getPriority() + ", Status: " + status + " on " + sdf.format(new Date()));
}

@Override
public String getStatus() { return status; }

public int getVoteCount() { return voteCount; }
public List<String> getComments() { return comments; }

public String getPriority() {
    if (voteCount <= 1) return "Low";
    else if (voteCount <= 5) return "Medium";
    else return "High";
}
}

class Proposal implements Engageable, Trackable, Notifiable {
    private static int proposalIdCounter = 1;
    private static List<String> proposals = new ArrayList<>();
    private String description;
    private Address location;
    private static List<String> comments = new ArrayList<>();
    private int voteCount = 0;
    private String status = "Not Reviewed";
    private String reportedBy;
    private Date reportedTime;

    class CommentManager {

```

```

    private void logComment(String comment) {
        System.out.println("Logging comment internally for Proposal " + proposalIdCounter);
    }
}

public Proposal(String description, Address location, String reportedBy) {
    this.description = description;
    this.location = location;
    this.reportedBy = reportedBy;
    this.reportedTime = new Date();
    proposals.add(description + " in " + location.getState() + ", " + location.getDistrict() + ", " +
location.getLocality());
    new CommentManager().logComment("Initial comment setup");
}

public static int getNextId() { return proposalIdCounter++; }
public static List<String> getProposals() { return proposals; }
public String getReportedBy() { return reportedBy; }
public Date getReportedTime() { return reportedTime; }
public Address getLocation() { return location; }
public void setDescription(String newDescription) {
    this.description = newDescription;
    int index = proposals.indexOf(getProposals().get(proposals.indexOf(this.description + " in "
+ location.getState() + ", " + location.getDistrict() + ", " + location.getLocality())));
    if (index >= 0) {
        proposals.set(index, newDescription + " in " + location.getState() + ", " +
location.getDistrict() + ", " + location.getLocality());
    }
}

@Override
public void addComment(String comment) throws IllegalArgumentException {
    if (comment == null || comment.trim().isEmpty()) {
        throw new IllegalArgumentException("Comment cannot be empty.");
    }
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    String timestampedComment = comment + " on " + sdf.format(new Date());
    comments.add(timestampedComment);
    System.out.println("Comment added to Proposal " + proposalIdCounter + ": " + comment +
" on " + sdf.format(new Date()));
}

@Override

```

```

public void addComment(String comment, String author) throws IllegalArgumentException {
    if (comment == null || comment.trim().isEmpty()) {
        throw new IllegalArgumentException("Comment cannot be empty.");
    }
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    String timestampedComment = comment + " by " + author + " on " + sdf.format(new
Date());
    comments.add(timestampedComment);
    System.out.println("Comment added to Proposal " + proposalIdCounter + ": " + comment +
" on " + sdf.format(new Date()));
}

public void addVote(int votes) {
    this.voteCount += votes;
    updatePriority();
}

private void updatePriority() {
    if (voteCount <= 1) {
        status = "Not Reviewed";
    } else if (voteCount <= 5) {
        status = "Under Review";
    } else {
        status = "Reviewed";
    }
}

@Override
public void updateStatus(String status) throws InvalidPriorityException {
    if (!status.matches("Low|Medium|High")) {
        throw new InvalidPriorityException("Invalid priority value. Use Low, Medium, or High.");
    }
    this.status = status;
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    int displayId = proposals.indexOf(description + " in " + location.getState() + ", " +
location.getDistrict() + ", " + location.getLocality()) + 1;
    System.out.println("Proposal " + displayId + " updated: " + description + " - Priority: " +
getPriority() + ", Status: " + status + " on " + sdf.format(new Date()));
}

@Override
public String getStatus() { return status; }

```

```

public int getVoteCount() { return voteCount; }
public List<String> getComments() { return comments; }

public String getPriority() {
    if (voteCount <= 1) return "Low";
    else if (voteCount <= 5) return "Medium";
    else return "High";
}

@Override
public void notifyUser(String message) {
    System.out.println("Notification for Proposal: " + message);
}
}

class Citizen extends RegisteredUser implements Notifiable {
    private static int userIdCounter = 100;
    private int userId;
    private static List<Citizen> registeredUsers = new ArrayList<>();
    private Set<Integer> votedIssues = new HashSet<>();
    private Set<Integer> votedProposals = new HashSet<>();
    private boolean isAuthority = false;
    private String gender;

    public Citizen() {
        super("DefaultUser", 0, "", "", "", new Address("", "", ""), "");
        this.userId = userIdCounter++;
        this.isAuthority = false;
        this.gender = "";
        registeredUsers.add(this);
    }

    public Citizen(String name, String password) {
        super(name, 0, "", "", "", new Address("", "", ""), password);
        this.userId = userIdCounter++;
        this.isAuthority = false;
        this.gender = "";
        registeredUsers.add(this);
    }

    public Citizen(String name, int age, String email, String aadharNo, String mobileNo, Address
address, String password, boolean isAuthority, String gender) {
        super(name, age, email, aadharNo, mobileNo, address, password);

```

```

    this.userId = userIdCounter++;
    this.isAuthority = isAuthority;
    this.gender = gender;
    registeredUsers.add(this);
}

public void voteIssue(Issue issue, List<Issue> issuesList) {
    if (!hasVotedIssue(issue.getNextId())) {
        issue.addVote(1);
        votedIssues.add(issue.getNextId());
        SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
        System.out.println("Vote recorded for Issue " + (issuesList.indexOf(issue) + 1) + " by " +
            getName() +
                ". New vote count: " + issue.getVoteCount() + " on " + sdf.format(new
            Date()));
    } else {
        System.out.println("You have already voted for this issue.");
    }
}

public void voteProposal(Proposal proposal, List<Proposal> proposalsList) {
    if (!hasVotedProposal(proposal.getNextId())) {
        proposal.addVote(1);
        votedProposals.add(proposal.getNextId());
        SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
        System.out.println("Vote recorded for Proposal " + (proposalsList.indexOf(proposal) + 1)
            + " by " + getName() +
                ". New vote count: " + proposal.getVoteCount() + " on " + sdf.format(new
            Date()));
    } else {
        System.out.println("You have already voted for this proposal.");
    }
}

public boolean isAuthority() { return isAuthority; }

public static Citizen authenticate(String name, String password) {
    for (Citizen user : registeredUsers) {
        if (user.getName().equals(name) && user.getPassword().equals(password)) {
            return user;
        }
    }
    return null;
}

```

```

    }

    public boolean hasVotedIssue(int issueId) {
        return votedIssues.contains(issueId);
    }

    public boolean hasVotedProposal(int proposalId) {
        return votedProposals.contains(proposalId);
    }

    @Override
    public void authenticate() {
        // Already handled by authenticate method
    }

    @Override
    public void notifyUser(String message) {
        System.out.println("Notification for Citizen: " + message);
    }

    public void editIssue(Issue issue, String newDescription, List<Issue> issuesList) {
        if (issuesList.contains(issue)) {
            issue.setDescription(newDescription);
            SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
            System.out.println("Issue " + (issuesList.indexOf(issue) + 1) + " edited to: " +
newDescription + " on " + sdf.format(new Date()));
        } else {
            System.out.println("Issue not found.");
        }
    }

    public String getGender() { return gender; }
    public int getUserId() { return userId; }
}

public class CivicPulse {
    static Scanner scanner = new Scanner(System.in);
    static List<Issue> issuesList = new ArrayList<>();
    static List<Proposal> proposalsList = new ArrayList<>();
    static Citizen currentCitizen;

    public static boolean isRegistered() {
        if (currentCitizen != null) return true;

```



```
System.out.println("Welcome to CIVIC PULSE - A Citizen Engagement & Participatory
Govt. Portal");
System.out.println("-----");
System.out.println("Citizen Voice, Govt. Action");
System.out.println("-----");
System.out.println("Please register to access the portal.");
System.out.println("1. Register");
System.out.println("2. Exit");
System.out.print("Enter your choice (1-2): ");
int choice;
try {
    choice = Integer.parseInt(scanner.nextLine());
} catch (NumberFormatException e) {
    System.out.println("Invalid choice. Please enter 1 or 2.");
    return false;
}
if (choice == 1) {
    System.out.print("Name: ");
    String name = scanner.nextLine().trim();
    System.out.print("Gender: ");
    String gender = scanner.nextLine().trim();
    int age;
    try {
        System.out.print("Age: ");
        age = Integer.parseInt(scanner.nextLine());
        if (age < 18) throw new IllegalArgumentException("Age must be 18 or above.");
    } catch (NumberFormatException e) {
        System.out.println("[Validation: " + e.getMessage() + "]");
        return false;
    } catch (IllegalArgumentException e) {
        System.out.println("[Validation: " + e.getMessage() + "]");
        return false;
    }
    System.out.print("Email: ");
    String email = scanner.nextLine().trim();
    System.out.print("Aadhar Number: ");
    String aadharNo = scanner.nextLine().trim();
    if (!aadharNo.matches("\\d{12}")) {
        System.out.println("[Validation: Aadhar number must be 12 digits.]");
        return false;
    }
    System.out.print("Mobile Number: ");
    String mobileNo = scanner.nextLine().trim();
```

```

if (!mobileNo.matches("\\d{10}")) {
    System.out.println("[Validation: Mobile number must be 10 digits.]");
    return false;
}
System.out.print("Enter State: ");
String state = scanner.nextLine().trim();
System.out.print("Enter District: ");
String district = scanner.nextLine().trim();
System.out.print("Enter Local Area: ");
String locality = scanner.nextLine().trim();
Address address = new Address(state, district, locality);
System.out.print("Set your password: ");
String password = scanner.nextLine().trim();
currentCitizen = new Citizen(name, age, email, aadharNo, mobileNo, address,
password, false, gender);
System.out.print("Submit - Yes/No: ");
String submit = scanner.nextLine().trim().toLowerCase();
if (submit.equals("yes")) {
    currentCitizen.notifyUser("Registration successful for " + name);
    System.out.println("You have Successfully Registered.");
    System.out.println("Welcome " + name + " !!");
    return true;
}
}
return false;
}

public static void viewDashboard() {
    System.out.println("--- Dashboard ---");
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    System.out.println("Dashboard Updated on " + sdf.format(new Date()));
    System.out.println("-----");
    if (!issuesList.isEmpty()) {
        System.out.println("Issues Reported:");
        for (int i = 0; i < issuesList.size(); i++) {
            System.out.println((i + 1) + ". Issue " + (i + 1) + ": " + issuesList.get(i).getIssues().get(i)
+
            " (Votes: " + issuesList.get(i).getVoteCount() + ", Priority: " +
issuesList.get(i).getPriority() +
            ", Status: " + issuesList.get(i).getStatus() + ")");
        }
    } else {
        System.out.println("No Issues Reported.");
    }
}

```

```

    }
    System.out.println("-----");
    if (!proposalsList.isEmpty()) {
        System.out.println("Proposals Submitted:");
        for (int i = 0; i < proposalsList.size(); i++) {
            System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
                " (Votes: " + proposalsList.get(i).getVoteCount() + ", Priority: " +
proposalsList.get(i).getPriority() +
                ", Status: " + proposalsList.get(i).getStatus() + ")");
        }
    } else {
        System.out.println("No Proposals Submitted.");
    }
    System.out.println("-----");
}

public static void generateReport() {
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    System.out.println("=== CIVIC PULSE - Comprehensive Report ===");
    System.out.println("Report Generated on: " + sdf.format(new Date()));
    System.out.println("-----");

    // Registration Details
    if (currentCitizen != null) {
        System.out.println("*** Registration Details ***");
        System.out.println("Name: " + currentCitizen.getName());
        System.out.println("Gender: " + currentCitizen.getGender());
        System.out.println("Age: " + currentCitizen.getAge());
        System.out.println("Email: " + currentCitizen.getEmail());
        System.out.println("Aadhar Number: " + currentCitizen.getAadharNo());
        System.out.println("Mobile Number: " + currentCitizen.getMobileNo());
        System.out.println("Address: " + currentCitizen.getAddress().getState() + ", " +
            currentCitizen.getAddress().getDistrict() + ", " +
            currentCitizen.getAddress().getLocality());
        System.out.println("Registration Date: " + sdf.format(new
Date(currentCitizen.getUserId() - 100))); // Approximate
        System.out.println("-----");
    } else {
        System.out.println("*** No Registered User Available ***");
        System.out.println("-----");
    }
}

```

```
// Issues Reported
System.out.println("*** Issues Reported ***");
if (!issuesList.isEmpty()) {
    for (int i = 0; i < issuesList.size(); i++) {
        Issue issue = issuesList.get(i);
        System.out.println((i + 1) + ". Issue: " + issue.getIssues().get(i));
        System.out.println("  Reported by: " + issue.getReportedBy());
        System.out.println("  Reported on: " + sdf.format(issue.getReportedTime()));
        System.out.println("  Vote Count: " + issue.getVoteCount());
        System.out.println("  Comments: " + (issue.getComments().isEmpty() ? "None" :
String.join(", ", issue.getComments())));
        System.out.println("  Priority: " + issue.getPriority());
        System.out.println("  Status: " + issue.getStatus());
        System.out.println("  -----");
    }
} else {
    System.out.println("No Issues Reported.");
    System.out.println("-----");
}

// Proposals Submitted
System.out.println("*** Proposals Submitted ***");
if (!proposalsList.isEmpty()) {
    for (int i = 0; i < proposalsList.size(); i++) {
        Proposal proposal = proposalsList.get(i);
        System.out.println((i + 1) + ". Proposal: " + proposal.getProposals().get(i));
        System.out.println("  Reported by: " + proposal.getReportedBy());
        System.out.println("  Reported on: " + sdf.format(proposal.getReportedTime()));
        System.out.println("  Vote Count: " + proposal.getVoteCount());
        System.out.println("  Comments: " + (proposal.getComments().isEmpty() ? "None" :
String.join(", ", proposal.getComments())));
        System.out.println("  Priority: " + proposal.getPriority());
        System.out.println("  Status: " + proposal.getStatus());
        System.out.println("  -----");
    }
} else {
    System.out.println("No Proposals Submitted.");
    System.out.println("-----");
}

// Review and Prioritization Status
System.out.println("*** Review and Prioritization Status ***");
boolean hasReviewed = false;
```

```

    for (Issue issue : issuesList) {
        if (!issue.getStatus().equals("Not Reviewed")) {
            System.out.println("Issue: " + issue.getIssues().get(issuesList.indexOf(issue)) + " -
Status: " + issue.getStatus());
            hasReviewed = true;
        }
    }
    for (Proposal proposal : proposalsList) {
        if (!proposal.getStatus().equals("Not Reviewed")) {
            System.out.println("Proposal: " +
proposal.getProposals().get(proposalsList.indexOf(proposal)) + " - Status: " +
proposal.getStatus());
            hasReviewed = true;
        }
    }
    if (!hasReviewed) {
        System.out.println("No items have been reviewed or prioritized yet.");
    }
    System.out.println("-----");

    // Dashboard Display
    System.out.println("*** Current Dashboard Snapshot ***");
    viewDashboard();
    System.out.println("-----");
    System.out.println("=== End of Report ===");
}

public static void main(String[] args) {
    if (!isRegistered()) {
        System.out.println("Thank You for Visiting our Site");
        return;
    }

    while (true) {
        System.out.println("Please select an action:");
        System.out.println("1. Report Issue (Citizen)");
        System.out.println("2. Submit Proposal (Citizen)");
        if (!issuesList.isEmpty() || !proposalsList.isEmpty()) {
            System.out.println("3. Vote on Issue/Proposal (Citizen)");
            System.out.println("4. Comment on Issue/Proposal (Citizen)");
            System.out.println("5. Review and Prioritize (Authority)");
            System.out.println("7. View Dashboard");
            System.out.println("8. Generate Report");
        }
    }
}

```

```

    }
    System.out.println("10. Exit");
    System.out.print("Enter your choice (1-10): ");

    int choice;
    try {
        choice = Integer.parseInt(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Invalid choice. Please enter 1-10.");
        continue;
    }

    if (choice == 1) {
        System.out.println("--- Report Issue ---");
        System.out.print("Enter Description: ");
        String description = scanner.nextLine().trim();
        System.out.print("Enter State: ");
        String state = scanner.nextLine().trim();
        System.out.print("Enter District: ");
        String district = scanner.nextLine().trim();
        System.out.print("Enter Local Area: ");
        String locality = scanner.nextLine().trim();
        Address address = new Address(state, district, locality);
        if (currentCitizen != null) {
            Issue issue = new Issue(description, address, currentCitizen.getName());
            issuesList.add(issue);
            SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
            System.out.println("Issue " + issue.getNextId() + " reported successfully: " +
description +
                " in " + state + ", " + district + ", " + locality + " on " + sdf.format(new
Date()));
            System.out.print("Submit - Yes/No: ");
            String submit = scanner.nextLine().trim().toLowerCase();
            if (!submit.equals("yes")) {
                System.out.println("Action cancelled");
            }
        } else {
            System.out.println("No registered user to report issue. Please register.");
        }
    } else if (choice == 2) {
        System.out.println("--- Submit Proposal ---");
        System.out.print("Enter Description: ");
        String description = scanner.nextLine().trim();
    }

```

```

System.out.print("Enter State: ");
String state = scanner.nextLine().trim();
System.out.print("Enter District: ");
String district = scanner.nextLine().trim();
System.out.print("Enter Local Area: ");
String locality = scanner.nextLine().trim();
Address address = new Address(state, district, locality);
if (currentCitizen != null) {
    Proposal proposal = new Proposal(description, address, currentCitizen.getName());
    proposalsList.add(proposal);
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy 'at' hh:mm a");
    System.out.println("Proposal " + proposal.getNextId() + " submitted successfully: "
+ description +
        " in " + state + ", " + district + ", " + locality + " on " + sdf.format(new
Date()));
    System.out.print("Submit - Yes/No: ");
    String submit = scanner.nextLine().trim().toLowerCase();
    if (!submit.equals("yes")) {
        System.out.println("Action cancelled");
    }
} else {
    System.out.println("No registered user to submit proposal. Please register.");
}
} else if (choice == 3 && (!issuesList.isEmpty() || !proposalsList.isEmpty())) {
    System.out.println("--- Vote on Issue/Proposal ---");
    System.out.print("Enter your name: ");
    String name = scanner.nextLine().trim();
    System.out.print("Enter your password: ");
    String password = scanner.nextLine().trim();
    Citizen authenticatedUser = Citizen.authenticate(name, password);
    if (authenticatedUser != null) {
        System.out.println("Authentication successful.");
        System.out.println("Select type to vote on:");
        System.out.println("1. Issue Reported");
        System.out.println("2. Proposal Submission");
        System.out.print("Enter your choice (1-2): ");
        int voteType = Integer.parseInt(scanner.nextLine());
        if (voteType == 1 && !issuesList.isEmpty()) {
            System.out.println("--- Vote on Issue ---");
            for (int i = 0; i < issuesList.size(); i++) {
                System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +
                    " (Votes: " + issuesList.get(i).getVoteCount() + ")");

```

```

    }
    System.out.print("Enter Issue ID to vote (1-" + issuesList.size() + "): ");
    int issueId = Integer.parseInt(scanner.nextLine()) - 1;
    if (issueId >= 0 && issueId < issuesList.size() &&
        !authenticatedUser.hasVotedIssue(issuesList.get(issueId).getNextId())) {
        authenticatedUser.voteIssue(issuesList.get(issueId), issuesList);
        System.out.print("Submit - Yes/No: ");
        String submit = scanner.nextLine().trim().toLowerCase();
        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
    } else {
        System.out.println("Invalid Issue ID or you have already voted.");
    }
} else if (voteType == 2 && !proposalsList.isEmpty()) {
    System.out.println("--- Vote on Proposal ---");
    for (int i = 0; i < proposalsList.size(); i++) {
        System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
            proposalsList.get(i).getProposals().get(i) +
            " (Votes: " + proposalsList.get(i).getVoteCount() + ")");
    }
    System.out.print("Enter Proposal ID to vote (1-" + proposalsList.size() + "): ");
    int proposalId = Integer.parseInt(scanner.nextLine()) - 1;
    if (proposalId >= 0 && proposalId < proposalsList.size() &&
        !authenticatedUser.hasVotedProposal(proposalsList.get(proposalId).getNextId())) {
        authenticatedUser.voteProposal(proposalsList.get(proposalId), proposalsList);
        System.out.print("Submit - Yes/No: ");
        String submit = scanner.nextLine().trim().toLowerCase();
        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
    } else {
        System.out.println("Invalid Proposal ID or you have already voted.");
    }
} else {
    System.out.println("Invalid choice or no items of that type available.");
}
} else {
    System.out.println("Incorrect username/password or You have not registered yet.");
    System.out.println("1. Re-enter username and password");
    System.out.println("2. Register Yourself");
    System.out.print("Enter your choice (1-2): ");

```



```

int authChoice = Integer.parseInt(scanner.nextLine());
if (authChoice == 1) {
    System.out.print("Enter your name: ");
    String reName = scanner.nextLine().trim();
    System.out.print("Enter your password: ");
    String rePassword = scanner.nextLine().trim();
    Citizen reAuthenticated = Citizen.authenticate(reName, rePassword);
    if (reAuthenticated != null) {
        authenticatedUser = reAuthenticated;
        System.out.println("Authentication successful.");
        System.out.println("Select type to vote on:");
        System.out.println("1. Issue Reported");
        System.out.println("2. Proposal Submission");
        System.out.print("Enter your choice (1-2): ");
        int voteType = Integer.parseInt(scanner.nextLine());
        if (voteType == 1 && !issuesList.isEmpty()) {
            System.out.println("--- Vote on Issue ---");
            for (int i = 0; i < issuesList.size(); i++) {
                System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +
                " (Votes: " + issuesList.get(i).getVoteCount() + ")");
            }
            System.out.print("Enter Issue ID to vote (1-" + issuesList.size() + "): ");
            int issuelid = Integer.parseInt(scanner.nextLine()) - 1;
            if (issuelid >= 0 && issuelid < issuesList.size() &&
!authenticatedUser.hasVotedIssue(issuesList.get(issuelid).getNextId())) {
                authenticatedUser.voteIssue(issuesList.get(issuelid), issuesList);
                System.out.print("Submit - Yes/No: ");
                String submit = scanner.nextLine().trim().toLowerCase();
                if (!submit.equals("yes")) {
                    System.out.println("Action cancelled");
                }
            } else {
                System.out.println("Invalid Issue ID or you have already voted.");
            }
        } else if (voteType == 2 && !proposalsList.isEmpty()) {
            System.out.println("--- Vote on Proposal ---");
            for (int i = 0; i < proposalsList.size(); i++) {
                System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
                " (Votes: " + proposalsList.get(i).getVoteCount() + ")");
            }
        }
    }
}

```

```

        System.out.print("Enter Proposal ID to vote (1-" + proposalsList.size() + "):
    ");

    int proposalId = Integer.parseInt(scanner.nextLine()) - 1;
    if (proposalId >= 0 && proposalId < proposalsList.size() &&
!authenticatedUser.hasVotedProposal(proposalsList.get(proposalId).getNextId())) {
        authenticatedUser.voteProposal(proposalsList.get(proposalId),
proposalsList);

        System.out.print("Submit - Yes/No: ");
        String submit = scanner.nextLine().trim().toLowerCase();
        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
        } else {
            System.out.println("Invalid Proposal ID or you have already voted.");
        }
        } else {
            System.out.println("Invalid choice or no items of that type available.");
        }
        } else {
            System.out.println("Incorrect username/password. Action cancelled.");
        }
    }
} else if (authChoice == 2) {
    currentCitizen = null;
    boolean registered = false;
    while (!registered) {
        registered = isRegistered();
        if (registered) {
            System.out.println("Registration successful. Please try voting again.");
            System.out.print("Enter your name: ");
            String reName = scanner.nextLine().trim();
            System.out.print("Enter your password: ");
            String rePassword = scanner.nextLine().trim();
            authenticatedUser = Citizen.authenticate(reName, rePassword);
            if (authenticatedUser != null) {
                System.out.println("Authentication successful.");
                System.out.println("Select type to vote on:");
                System.out.println("1. Issue Reported");
                System.out.println("2. Proposal Submission");
                System.out.print("Enter your choice (1-2): ");
                int voteType = Integer.parseInt(scanner.nextLine());
                if (voteType == 1 && !issuesList.isEmpty()) {
                    System.out.println("--- Vote on Issue ---");

```

```

    for (int i = 0; i < issuesList.size(); i++) {
        System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +
            " (Votes: " + issuesList.get(i).getVoteCount() + ")");
    }
    System.out.print("Enter Issue ID to vote (1-" + issuesList.size() + "): ");
    int issuelid = Integer.parseInt(scanner.nextLine()) - 1;
    if (issuelid >= 0 && issuelid < issuesList.size() &&
!authenticatedUser.hasVotedIssue(issuesList.get(issuelid).getNextId())) {
        authenticatedUser.voteIssue(issuesList.get(issuelid), issuesList);
        System.out.print("Submit - Yes/No: ");
        String submit = scanner.nextLine().trim().toLowerCase();
        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
    } else {
        System.out.println("Invalid Issue ID or you have already voted.");
    }
} else if (voteType == 2 && !proposalsList.isEmpty()) {
    System.out.println("--- Vote on Proposal ---");
    for (int i = 0; i < proposalsList.size(); i++) {
        System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
            " (Votes: " + proposalsList.get(i).getVoteCount() + ")");
    }
    System.out.print("Enter Proposal ID to vote (1-" + proposalsList.size()
+ "): ");
    int proposalid = Integer.parseInt(scanner.nextLine()) - 1;
    if (proposalid >= 0 && proposalid < proposalsList.size() &&
!authenticatedUser.hasVotedProposal(proposalsList.get(proposalid).getNextId())) {
        authenticatedUser.voteProposal(proposalsList.get(proposalid),
proposalsList);

        System.out.print("Submit - Yes/No: ");
        String submit = scanner.nextLine().trim().toLowerCase();
        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
    } else {
        System.out.println("Invalid Proposal ID or you have already voted.");
    }
} else {

```

```

        System.out.println("Invalid choice or no items of that type available.");
    }
    } else {
        System.out.println("Authentication failed after registration. Please try
again.");
    }
    break;
    } else {
        System.out.println("Registration failed or cancelled. Please try again or
exit.");
    }
    }
    } else {
        System.out.println("Invalid choice. Action cancelled.");
    }
    }
    } else if (choice == 4 && (!issuesList.isEmpty() || !proposalsList.isEmpty())) {
        if (issuesList.isEmpty() && !proposalsList.isEmpty()) {
            System.out.println("--- Comment on Proposal ---");
            for (int i = 0; i < proposalsList.size(); i++) {
                System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
                " (Votes: " + proposalsList.get(i).getVoteCount() + ")");
            }
            System.out.print("Enter Proposal ID to comment (1-" + proposalsList.size() + "): ");
            int proposalId = Integer.parseInt(scanner.nextLine()) - 1;
            if (proposalId >= 0 && proposalId < proposalsList.size()) {
                System.out.print("Enter your comment: ");
                String comment = scanner.nextLine();
                if (currentCitizen != null) {
                    proposalsList.get(proposalId).addComment(comment,
currentCitizen.getName());
                } else {
                    proposalsList.get(proposalId).addComment(comment);
                }
                System.out.print("Submit - Yes/No: ");
                String submit = scanner.nextLine().trim().toLowerCase();
                if (!submit.equals("yes")) {
                    System.out.println("Action cancelled");
                }
            } else {
                System.out.println("Invalid Proposal ID.");
            }
        }
    }

```

```

    } else if (!issuesList.isEmpty() && proposalsList.isEmpty()) {
        System.out.println("--- Comment on Issue ---");
        for (int i = 0; i < issuesList.size(); i++) {
            System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +
                " (Votes: " + issuesList.get(i).getVoteCount() + ")");
        }
        System.out.print("Enter Issue ID to comment (1-" + issuesList.size() + "): ");
        int issuelid = Integer.parseInt(scanner.nextLine()) - 1;
        if (issuelid >= 0 && issuelid < issuesList.size()) {
            System.out.print("Enter your comment: ");
            String comment = scanner.nextLine();
            if (currentCitizen != null) {
                issuesList.get(issuelid).addComment(comment, currentCitizen.getName());
            } else {
                issuesList.get(issuelid).addComment(comment);
            }
            System.out.print("Submit - Yes/No: ");
            String submit = scanner.nextLine().trim().toLowerCase();
            if (!submit.equals("yes")) {
                System.out.println("Action cancelled");
            }
        } else {
            System.out.println("Invalid Issue ID.");
        }
    } else {
        System.out.println("Select type to comment on:");
        System.out.println("1. Issue Reported");
        System.out.println("2. Proposal Submission");
        System.out.print("Enter your choice (1-2): ");
        int commentType = Integer.parseInt(scanner.nextLine());
        if (commentType == 1) {
            System.out.println("--- Comment on Issue ---");
            for (int i = 0; i < issuesList.size(); i++) {
                System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +
                    " (Votes: " + issuesList.get(i).getVoteCount() + ")");
            }
            System.out.print("Enter Issue ID to comment (1-" + issuesList.size() + "): ");
            int issuelid = Integer.parseInt(scanner.nextLine()) - 1;
            if (issuelid >= 0 && issuelid < issuesList.size()) {
                System.out.print("Enter your comment: ");
                String comment = scanner.nextLine();

```

```

    if (currentCitizen != null) {
        issuesList.get(issueId).addComment(comment, currentCitizen.getName());
    } else {
        issuesList.get(issueId).addComment(comment);
    }
    System.out.print("Submit - Yes/No: ");
    String submit = scanner.nextLine().trim().toLowerCase();
    if (!submit.equals("yes")) {
        System.out.println("Action cancelled");
    }
} else {
    System.out.println("Invalid Issue ID.");
}
} else if (commentType == 2) {
    System.out.println("--- Comment on Proposal ---");
    for (int i = 0; i < proposalsList.size(); i++) {
        System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
        " (Votes: " + proposalsList.get(i).getVoteCount() + ")");
    }
    System.out.print("Enter Proposal ID to comment (1-" + proposalsList.size() + "):
");
    int proposalId = Integer.parseInt(scanner.nextLine()) - 1;
    if (proposalId >= 0 && proposalId < proposalsList.size()) {
        System.out.print("Enter your comment: ");
        String comment = scanner.nextLine();
        if (currentCitizen != null) {
            proposalsList.get(proposalId).addComment(comment,
currentCitizen.getName());
        } else {
            proposalsList.get(proposalId).addComment(comment);
        }
        System.out.print("Submit - Yes/No: ");
        String submit = scanner.nextLine().trim().toLowerCase();
        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
    } else {
        System.out.println("Invalid Proposal ID.");
    }
} else {
    System.out.println("Invalid choice. Please enter 1 or 2.");
}
}

```

```

    }
} else if (choice == 5 && (!issuesList.isEmpty() || !proposalsList.isEmpty())) {
    System.out.print("[Authority login check: Enter role (Citizen/Authority): ");
    String role = scanner.nextLine().trim();
    if (role.equalsIgnoreCase("Authority")) {
        try {
            if (issuesList.isEmpty() && !proposalsList.isEmpty()) {
                System.out.println("--- Review Proposals ---");
                for (int i = 0; i < proposalsList.size(); i++) {
                    System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
                    " (Votes: " + proposalsList.get(i).getVoteCount() + ", Priority: " +
proposalsList.get(i).getPriority() +
                    ", Status: " + proposalsList.get(i).getStatus() + ")");
                }
                System.out.print("Enter Proposal ID to review (1-" + proposalsList.size() + "):
");
                int proposalId = Integer.parseInt(scanner.nextLine()) - 1;
                if (proposalId >= 0 && proposalId < proposalsList.size()) {
                    System.out.println("Current Priority: " +
proposalsList.get(proposalId).getPriority() +
                    " (based on " + proposalsList.get(proposalId).getVoteCount() + "
votes)");
                    System.out.print("Confirm or adjust priority (Low/Medium/High): ");
                    String priority = scanner.nextLine().trim();
                    if (priority.matches("Low|Medium|High")) {
                        proposalsList.get(proposalId).updateStatus(priority);
                        System.out.print("Submit - Yes/No: ");
                        String submit = scanner.nextLine().trim().toLowerCase();
                        if (!submit.equals("yes")) {
                            System.out.println("Action cancelled");
                        }
                    } else {
                        System.out.println("Invalid priority. Action cancelled.");
                    }
                } else {
                    System.out.println("Invalid Proposal ID.");
                }
            } else if (!issuesList.isEmpty() && proposalsList.isEmpty()) {
                System.out.println("--- Review Issues ---");
                for (int i = 0; i < issuesList.size(); i++) {
                    System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +

```

```

        " (Votes: " + issuesList.get(i).getVoteCount() + ", Priority: " +
issuesList.get(i).getPriority() +
        ", Status: " + issuesList.get(i).getStatus() + ")");
    }
    System.out.print("Enter Issue ID to review (1-" + issuesList.size() + "): ");
    int issuelid = Integer.parseInt(scanner.nextLine()) - 1;
    if (issuelid >= 0 && issuelid < issuesList.size()) {
        System.out.println("Current Priority: " + issuesList.get(issuelid).getPriority()
+
        " (based on " + issuesList.get(issuelid).getVoteCount() + "
votes)");

        System.out.print("Confirm or adjust priority (Low/Medium/High): ");
        String priority = scanner.nextLine().trim();
        if (priority.matches("Low|Medium|High")) {
            issuesList.get(issuelid).updateStatus(priority);
            System.out.print("Submit - Yes/No: ");
            String submit = scanner.nextLine().trim().toLowerCase();
            if (!submit.equals("yes")) {
                System.out.println("Action cancelled");
            }
        } else {
            System.out.println("Invalid priority. Action cancelled.");
        }
    } else {
        System.out.println("Invalid Issue ID.");
    }
} else {
    System.out.println("Select type to review:");
    System.out.println("1. Issue Reported");
    System.out.println("2. Proposal Submission");
    System.out.print("Enter your choice (1-2): ");
    int reviewType = Integer.parseInt(scanner.nextLine());
    if (reviewType == 1) {
        System.out.println("--- Review Issues ---");
        for (int i = 0; i < issuesList.size(); i++) {
            System.out.println((i + 1) + ". Issue " + (i + 1) + ": " +
issuesList.get(i).getIssues().get(i) +
            " (Votes: " + issuesList.get(i).getVoteCount() + ", Priority: " +
issuesList.get(i).getPriority() +
            ", Status: " + issuesList.get(i).getStatus() + ")");
        }
        System.out.print("Enter Issue ID to review (1-" + issuesList.size() + "): ");
        int issuelid = Integer.parseInt(scanner.nextLine()) - 1;

```



```

    if (issuelid >= 0 && issuelid < issuesList.size()) {
        System.out.println("Current Priority: " +
issuesList.get(issuelid).getPriority() +
            " (based on " + issuesList.get(issuelid).getVoteCount() + "
votes)");

        System.out.print("Confirm or adjust priority (Low/Medium/High): ");
        String priority = scanner.nextLine().trim();
        if (priority.matches("Low|Medium|High")) {
            issuesList.get(issuelid).updateStatus(priority);
            System.out.print("Submit - Yes/No: ");
            String submit = scanner.nextLine().trim().toLowerCase();
            if (!submit.equals("yes")) {
                System.out.println("Action cancelled");
            }
        } else {
            System.out.println("Invalid priority. Action cancelled.");
        }
    } else {
        System.out.println("Invalid Issue ID.");
    }
} else if (reviewType == 2) {
    System.out.println("--- Review Proposals ---");
    for (int i = 0; i < proposalsList.size(); i++) {
        System.out.println((i + 1) + ". Proposal " + (i + 1) + ": " +
proposalsList.get(i).getProposals().get(i) +
            " (Votes: " + proposalsList.get(i).getVoteCount() + ", Priority: "
+ proposalsList.get(i).getPriority() +
            ", Status: " + proposalsList.get(i).getStatus() + ")");
    }
    System.out.print("Enter Proposal ID to review (1-" + proposalsList.size() +
"): ");

    int proposalId = Integer.parseInt(scanner.nextLine()) - 1;
    if (proposalId >= 0 && proposalId < proposalsList.size()) {
        System.out.println("Current Priority: " +
proposalsList.get(proposalId).getPriority() +
            " (based on " + proposalsList.get(proposalId).getVoteCount()
+ " votes)");

        System.out.print("Confirm or adjust priority (Low/Medium/High): ");
        String priority = scanner.nextLine().trim();
        if (priority.matches("Low|Medium|High")) {
            proposalsList.get(proposalId).updateStatus(priority);
            System.out.print("Submit - Yes/No: ");
            String submit = scanner.nextLine().trim().toLowerCase();

```

```

        if (!submit.equals("yes")) {
            System.out.println("Action cancelled");
        }
    } else {
        System.out.println("Invalid priority. Action cancelled.");
    }
} else {
    System.out.println("Invalid Proposal ID.");
}
} else {
    System.out.println("Invalid choice. Please enter 1 or 2.");
}
}
} catch (NumberFormatException e) {
    System.out.println("Invalid ID. Please enter a number.");
} catch (InvalidPriorityException e) {
    System.out.println("[Validation: " + e.getMessage() + "]");
} catch (IllegalArgumentException e) {
    System.out.println("[Validation: " + e.getMessage() + "]");
}
}
} else if (choice == 7 && (!issuesList.isEmpty() || !proposalsList.isEmpty())) {
    viewDashboard();
} else if (choice == 8 && (!issuesList.isEmpty() || !proposalsList.isEmpty())) {
    generateReport();
} else if (choice == 10) {
    System.out.println("Thank You for Visiting our Site");
    break;
} else if ((choice == 5 || choice == 7 || choice == 8) && (issuesList.isEmpty() &&
proposalsList.isEmpty())) {
    System.out.println("Invalid choice. No issues or proposals available for review,
dashboard, or report. Please enter 1, 2, or 10.");
    System.out.print("Would you like to perform another action? (Yes/No): ");
    String anotherAction = scanner.nextLine().trim().toLowerCase();
    if (anotherAction.equals("no")) {
        System.out.println("Thank You for Visiting our Site");
        break;
    }
} else {
    System.out.println("Invalid choice. Please enter a valid option.");
}
}
scanner.close();

```

```
}  
}
```

## 9. Outputs

### Registration + Issue Reporting / Proposal Submission

```
C:\Myjava>java CivicPulse.java
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Ayush Kumar
Gender: Male
Age: 19
Email: ayush@gmail.com
Aadhar No. (12 digits): 437637839883
Mobile No.: 8756382909
Address - Country (India): State: Karnataka
District: Bangalore
Pin Code: 562157
Locality: CP Layout, Hunsmaranhalli
Submit - Yes/No: Yes
You have Successfully Registered.
Welcome Ayush Kumar !!
Please select an action:
1. Report Issue
2. Submit Proposal
3. Exit
Enter your choice (1-3): 1
--- Report Issue ---
Enter Description: Pothole on Main Road
Enter State: Karnataka
Enter District: Bangalore
Enter Local Area: Devanhalli
Submit - Yes/No: Yes
Issue 1 reported successfully: Pothole on Main Road in Karnataka, Bangalore, Devanhalli on 04/12/2025 at 02:37 AM
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Exit
```

```
Enter your choice (1-3): 2
--- Submit Proposal ---
Enter Description: Build a Park
Enter State: Jharkhand
Enter District: Koderma
Enter Local Area: Urwan
Submit - Yes/No: Yes
Proposal 1 submitted successfully: Build a Park in Jharkhand, Koderma, Urwan on 04/12/2025 at 02:38 AM
Would you like to perform another action? (Yes/No): No
Thank You for using Civic Pulse
```

### Vote for Reporting Issue / Proposal Submission

```
Please select an action:
1. Report Issue
2. Submit Proposal
4. Exit
Enter your choice (1-4): 1
--- Report Issue ---
Enter Description: Water Leakage
Enter State: West Bengal
Enter District: Bardhaman
Enter Local Area: Durgapur
Submit - Yes/No: Yes
Issue 1 reported successfully: Water Leakage in West Bengal, Bardhaman, Durgapur on 04/12/2025 at 03:14 AM
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Exit
Enter your choice (1-4): 2
--- Submit Proposal ---
Enter Description: Build a Library
Enter State: Bihar
Enter District: Gaya
Enter Local Area: RP Nagar
Submit - Yes/No: Yes
Proposal 1 submitted successfully: Build a Library in Bihar, Gaya, RP Nagar on 04/12/2025 at 03:16 AM
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Exit
Enter your choice (1-4): 1
--- Report Issue ---
Enter Description: Broken StreetLight
Enter State: Jharkhnad
Enter District: Giridih
Enter Local Area: Gandhi Road
Submit - Yes/No: Yes
Issue 2 reported successfully: Broken StreetLight in Jharkhnad, Giridih, Gandhi Road on 04/12/2025 at 03:20 AM
Would you like to perform another action? (Yes/No): Yes
```

```

Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Exit
Enter your choice (1-4): 2
--- Submit Proposal ---
Enter Description: Build a Primary School
Enter State: Punjab
Enter District: Amritsar
Enter Local Area: SP Marg
Submit - Yes/No: Yes
Proposal 2 submitted successfully: Build a Primary School in Punjab, Amritsar, SP Marg on 04/12/2025 at 03:22 AM
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Exit
Enter your choice (1-4): 3
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Leakage in West Bengal, Bardhaman, Durgapur
2. Issue 2: Broken StreetLight in Jharkhnad, Giridih, Gandhi Road
Enter Issue ID to vote (1-2): 2
Vote recorded for Issue 2. New vote count: 1 on 04/12/2025 at 03:22 AM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Exit
Enter your choice (1-4): 3
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Library in Bihar, Gaya, RP Nagar
2. Proposal 2: Build a Primary School in Punjab, Amritsar, SP Marg
Enter Proposal ID to vote (1-2): 1
Vote recorded for Proposal 1. New vote count: 1 on 04/12/2025 at 03:23 AM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): No
Thank You for using Civic Pulse

```

## Commenting on Issue Reported / Proposal Submittedn

```
c:\myjava>java -Duser.dir=.deser.java
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Muskan Kumari
Gender: Female
Age: 19
Email: muskan@gmail.com
Aadhar No. (12 digits): 635323738942
Mobile No.: 8725372678
Address - Country (India): State: Delhi
District: New Delhi
Pin Code: 657632
Locality: Green Park
Submit - Yes/No: Yes
You have Successfully Registered.
Welcome Muskan Kumari !!
Please select an action:
1. Report Issue
2. Submit Proposal
5. Exit
Enter your choice (1-5): 1
--- Report Issue ---
Enter Description: Supply Water Pipeline Leakage
Enter State: Delhi
Enter District: New Delhi
Enter Local Area: Rajendra Nagar
Submit - Yes/No: Yes
Issue 1 reported successfully: Supply Water Pipeline Leakage in Delhi, New Delhi, Rajendra Nagar on 04/12/2025 at 11:41 AM
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Comment on Issue/Proposal
5. Exit
```

```
Enter your choice (1-5): 2
--- Submit Proposal ---
Enter Description: Build a Water Tank
Enter State: Delhi
Enter District: New Delhi
Enter Local Area: Lalbagh
Submit - Yes/No: Yes
Proposal 1 submitted successfully: Build a Water Tank in Delhi, New Delhi, Lalbagh on 04/12/2025 at 11:42 AM
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Comment on Issue/Proposal
5. Exit
Enter your choice (1-5): 3
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Supply Water Pipeline Leakage in Delhi, New Delhi, Rajendra Nagar
Enter Issue ID to vote (1-1): 1
Vote recorded for Issue 1. New vote count: 1 on 04/12/2025 at 11:42 AM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Comment on Issue/Proposal
5. Exit
Enter your choice (1-5): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Comment on Proposal ---
1. Proposal 1: Build a Water Tank in Delhi, New Delhi, Lalbagh
Enter Proposal ID to comment (1-1): Yes, it's needed for public
Invalid ID. Please enter a number.
```

```
Would you like to perform another action? (Yes/No): yes
Please select an action:
1. Report Issue
2. Submit Proposal
3. Vote on Issue/Proposal
4. Comment on Issue/Proposal
5. Exit
Enter your choice (1-5): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Comment on Proposal ---
1. Proposal 1: Build a Water Tank in Delhi, New Delhi, Lalbagh
Enter Proposal ID to comment (1-1): 1
Enter your comment: Yes, it's needed for Local Public
Comment added to Proposal 2: Yes, it's needed for Local Public on 04/12/2025 at 11:44 AM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): No
Thank You for using Civic Pulse
```

**Voting Updated - One/Multiple Users can vote in one session (one user - one vote)**

```
Welcome to CIVIC PULSE – A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Ayush Kumar
Gender: Male
Age: 19
Email: ayush@gmail.com
Aadhar Number: 673528385289
Mobile Number: 7886684738
Enter State: Jharkhand
Enter District: Koderma
Enter Local Area: Urwan More
Set your password: 1098
Submit - Yes/No: Yes
You have Successfully Registered.
Welcome Ayush Kumar !!
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
6. Exit
Enter your choice (1-6): 1
--- Report Issue ---
Enter Description: Pothole on main road
Enter State: Jharkhand
Enter District: Koderma
Enter Local Area: Telaiya Dam Road
Issue 1 reported successfully: Pothole on main road in Jharkhand, Koderma, Telaiya Dam Road on 04/13/2025 at 06:51 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
-----
Enter your choice (1-6): 2
--- Submit Proposal ---
Enter Description: Build a park
Enter State: Bihar
Enter District: Patna
Enter Local Area: Gandhi Colony
Proposal 1 submitted successfully: Build a park in Bihar, Patna, Gandhi Colony on 04/13/2025 at 06:52 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
```



```
Enter your choice (1-6): 1
--- Report Issue ---
Enter Description: Broken Street Light
Enter State: Uttar Pradesh
Enter District: Prayagraj
Enter Local Area: Sangam Road
Issue 2 reported successfully: Broken Street Light in Uttar Pradesh, Prayagraj, Sangam Road on 04/13/2025 at 06:53 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 2
--- Submit Proposal ---
Enter Description: Build a Library
Enter State: Rajasthan
Enter District: Kota
Enter Local Area: PW Road
Proposal 2 submitted successfully: Build a Library in Rajasthan, Kota, PW Road on 04/13/2025 at 06:54 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 3
--- Vote on Issue/Proposal ---
Enter your name: Ayush Kum
Enter your password: 1098
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
```

```
Enter your choice (1-2): 1
Enter your name: Ayush Kumar
Enter your password: 1098
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Pothole on main road in Jharkhand, Koderma, Telaiya Dam Road (Votes: 0)
2. Issue 2: Broken Street Light in Uttar Pradesh, Prayagraj, Sangam Road (Votes: 0)
Enter Issue ID to vote (1-2): 1
Vote recorded for Issue 1. New vote count: 1 on 04/13/2025 at 06:55 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
```

```
Enter your choice (1-6): 3
--- Vote on Issue/Proposal ---
Enter your name: Ayush Kumar
Enter your password: 1098
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a park in Bihar, Patna, Gandhi Colony (Votes: 0)
2. Proposal 2: Build a Library in Rajasthan, Kota, PW Road (Votes: 0)
Enter Proposal ID to vote (1-2): 2
Vote recorded for Proposal 2. New vote count: 1 on 04/13/2025 at 06:56 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Comment on Issue ---
1. Issue 1: Pothole on main road in Jharkhand, Koderma, Telaiya Dam Road (Votes: 1)
2. Issue 2: Broken Street Light in Uttar Pradesh, Prayagraj, Sangam Road (Votes: 0)
Enter Issue ID to comment (1-2): 1
Enter your comment: It's Urgent due to rain
Comment added to Issue 6: It's Urgent due to rain on 04/13/2025 at 06:57 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
```

```
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Comment on Proposal ---
1. Proposal 1: Build a park in Bihar, Patna, Gandhi Colony (Votes: 0)
2. Proposal 2: Build a Library in Rajasthan, Kota, PW Road (Votes: 1)
Enter Proposal ID to comment (1-2): 2
Enter your comment: Yes, it's a need for students
Comment added to Proposal 6: Yes, it's a need for students on 04/13/2025 at 06:58 PM
Submit - Yes/No: Yes
```

```
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue      (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 3
--- Vote on Issue/Proposal ---
Enter your name: Sujal Giri
Enter your password: 1013
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 2
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Sujal Giri
Gender: Male
Age: 20
Email: suj@l@gmail.com
Aadhar Number: 452748263849
Mobile Number: 9347374738
Enter State: West Bengal
Enter District: Burdhan
Enter Local Area: Asansol
Set your password: 1013
Submit - Yes/No: Yes
You have Successfully Registered.
Welcome Sujal Giri !!
Registration successful. Please try voting again.
Enter your name: Sujal Giri
Enter your password: 1013
Authentication successful.
```

```
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Pothole on main road in Jharkhand, Koderma, Telaiya Dam Road (Votes: 1)
2. Issue 2: Broken Street Light in Uttar Pradesh, Prayagraj, Sangam Road (Votes: 0)
Enter Issue ID to vote (1-2): 1
Vote recorded for Issue 1. New vote count: 2 on 04/13/2025 at 07:00 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 3
--- Vote on Issue/Proposal ---
Enter your name: Sujal Giri
Enter your password: 1098
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 1
Enter your name: Sujal Giri
Enter your password: 1013
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a park in Bihar, Patna, Gandhi Colony (Votes: 0)
2. Proposal 2: Build a Library in Rajasthan, Kota, PW Road (Votes: 1)
Enter Proposal ID to vote (1-2): 2
Vote recorded for Proposal 2. New vote count: 2 on 04/13/2025 at 07:01 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes

Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
```

### Authority Access for Review and Prioritizing Purpose

```

Enter your choice (1-6): 5
[Authority login check: Enter role (Citizen/Authority): Citizen
Would you like to perform another action? (Yes/No): Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 5
[Authority login check: Enter role (Citizen/Authority): Authority
Select type to review:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Review Issues ---
1. Issue 1: Pothole on main road in Jharkhand, Koderma, Telaiya Dam Road (Votes: 2, Priority: Medium, Status: Under Review)
2. Issue 2: Broken Street Light in Uttar Pradesh, Prayagraj, Sangam Road (Votes: 0, Priority: Low, Status: Not Reviewed)
Enter Issue ID to review (1-2): 1
Current Priority: Medium (based on 2 votes)
Confirm or adjust priority (Low/Medium/High): High
Issue 1 updated: Pothole on main road - Priority: Medium, Status: High on 04/13/2025 at 07:03 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): Yes
Would you like to perform another action? (Yes/No): Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
6. Exit
Enter your choice (1-6): 5
[Authority login check: Enter role (Citizen/Authority): Authority
Select type to review:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Review Proposals ---
1. Proposal 1: Build a park in Bihar, Patna, Gandhi Colony (Votes: 0, Priority: Low, Status: Not Reviewed)
2. Proposal 2: Build a Library in Rajasthan, Kota, PW Road (Votes: 2, Priority: Medium, Status: Under Review)
Enter Proposal ID to review (1-2): 2
Current Priority: Medium (based on 2 votes)
Confirm or adjust priority (Low/Medium/High): Medium
Proposal 2 updated: Build a Library - Priority: Medium, Status: Medium on 04/13/2025 at 07:05 PM
Submit - Yes/No: Yes
Would you like to perform another action? (Yes/No): No
Thank You for Visiting our Site

```

## Final Output with all the Operations

```
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Amrit Kumar
Gender: Male
Age: 25
Email: amrit@gmail.com
Aadhar Number: 765246523781
Mobile Number: 9854766887
Enter State: Karnataka
Enter District: Bangalore
Enter Local Area: Yelanka
Set your password: 1013
Submit - Yes/No: Yes
Notification for Citizen: Registration successful for Amrit Kumar
You have Successfully Registered.
Welcome Amrit Kumar !!
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
10. Exit
Enter your choice (1-10): 1
--- Report Issue ---
Enter Description: Water Drainage
Enter State: Karnataka
Enter District: Bangalore
Enter Local Area: Kogilu Cross
Logging comment internally for Issue 1
Issue 1 reported successfully: Water Drainage in Karnataka, Bangalore, Kogilu Cross on 04/13/2025 at 09:51 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 2
--- Submit Proposal ---
Enter Description: Build a Water tank
```

```

Enter Description: Build a Water tank
Enter State: Bihar
Enter District: Gaya
Enter Local Area: PLA Road
Logging comment internally for Proposal 1
Proposal 1 submitted successfully: Build a Water tank in Bihar, Gaya, PLA Road on 04/13/2025 at 09:52 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Amrit Kum
Enter your password: 1013
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 1
Enter your name: Amrit Kumar
Enter your password: 1013
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 0)
Enter Issue ID to vote (1-1): 1
Vote recorded for Issue 1 by Amrit Kumar. New vote count: 1 on 04/13/2025 at 09:53 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Amrit Kumar
Enter your password: 1013
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 0)
Enter Proposal ID to vote (1-1): 1
Vote recorded for Proposal 1 by Amrit Kumar. New vote count: 1 on 04/13/2025 at 09:58 PM
Submit - Yes/No: Yes

```

```

Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit

```

```
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Vikram Kumar
Enter your password: 1243
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 2
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Vikram Kumar
Gender: Male
Age: 23
Email: vikram@gmail.com
Aadhar Number: 756376482363
Mobile Number: 8763732898
Enter State: Goa
Enter District: Panji
Enter Local Area: Vasco Da Gama
Set your password: 3223
Submit - Yes/No: Yes
Notification for Citizen: Registration successful for Vikram Kumar
You have Successfully Registered.
Welcome Vikram Kumar !!
Registration successful. Please try voting again.
Enter your name: Vikram Kumar
Enter your password: 3223
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 1)
Enter Issue ID to vote (1-1): 1
Vote recorded for Issue 1 by Vikram Kumar. New vote count: 2 on 04/13/2025 at 10:02 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 1
--- Report Issue ---
Enter Description: Poor road conditions
Enter State: Goa
Enter District: Panji
Enter Local Area: GR Road
```



```
Issue 8 reported successfully: Poor road conditions in Goa, Panji, GR Road on 04/13/2025 at 10:03 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 2
--- Submit Proposal ---
Enter Description: Installation of CCTV Cameras
Enter State: Delhi
Enter District: New Delhi
Enter Local Area: Green Circle
Logging comment internally for Proposal 5
Proposal 5 submitted successfully: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle on 04/13/2025 at 10:05 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Vikram Kumar
Enter your password: 3223
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 2)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 0)
Enter Issue ID to vote (1-2): 2
Vote recorded for Issue 2 by Vikram Kumar. New vote count: 1 on 04/13/2025 at 10:06 PM
Submit - Yes/No: Yes
Please select an action:
```

```

Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Vikram Kumar
Enter your password: 3223
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 1)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 0)
Enter Proposal ID to vote (1-2): 2
Vote recorded for Proposal 2 by Vikram Kumar. New vote count: 1 on 04/13/2025 at 10:06 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Sudhir Singh
Enter your password: 8907
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 2
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register

```

```

2. Exit
Enter your choice (1-2): 1
Name: Sudhir Singh
Gender: Male
Age: 27
Email: sudhir@gmail.com
Aadhar Number: 567384632728
Mobile Number: 9856546677
Enter State: Uttar Pradesh
Enter District: Lucknow
Enter Local Area: Chatori Gali
Set your password: 8907
Submit - Yes/No: Yes
Notification for Citizen: Registration successful for Sudhir Singh
You have Successfully Registered.
Welcome Sudhir Singh !!

```

```
Welcome Sudhir Singh !!
Registration successful. Please try voting again.
Enter your name: Sudhir Singh
Enter your password: 8907
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 2)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 1)
Enter Issue ID to vote (1-2): 1
Vote recorded for Issue 1 by Sudhir Singh. New vote count: 3 on 04/13/2025 at 10:11 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 2
--- Submit Proposal ---
Enter Description: Build a Local Community Park
Enter State: Uttar Pradesh
Enter District: Lucknow
Enter Local Area: Charbagh
Logging comment internally for Proposal 9
Proposal 9 submitted successfully: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh on 04/13/2025 at 10:12 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Sudhir Singh
Enter your password: 8907
Authentication successful.
```

```
Authentication Successful
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 3)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 1)
Enter Issue ID to vote (1-2): 2
Vote recorded for Issue 2 by Sudhir Singh. New vote count: 2 on 04/13/2025 at 10:13 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Sudhir Singh
Enter your password: 8907
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 1)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 1)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 0)
Enter Proposal ID to vote (1-3): 2
Vote recorded for Proposal 2 by Sudhir Singh. New vote count: 2 on 04/13/2025 at 10:14 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 1
--- Report Issue ---
```

```

--- Report Issue ---
Enter Description: Irregular Garbage Collection
Enter State: Uttar Pradesh
Enter District: Lucknow
Enter Local Area: Chatori Gali
Logging comment internally for Issue 18
Issue 18 reported successfully: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali on 04/13/2025 at 10:15 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Karan Kapoor
Enter your password: 5678
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 2
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit
Enter your choice (1-2): 1
Name: Karan Kapoor
Gender: Male
Age: 22
Email: karan@gmail.com
Aadhar Number: 745637328389
Mobile Number: 9836487482
Enter State: Haryana
Enter District: Hisar
Enter Local Area: Sector 13
Set your password: 6778
Submit - Yes/No: Yes
Notification for Citizen: Registration successful for Karan Kapoor
You have Successfully Registered.
Welcome Karan Kapoor !!

```

```

Welcome Karan Kapoor !!
Registration successful. Please try voting again.
Enter your name: Karan Kapoor
Enter your password: 6778
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 3)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 2)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 0)
Enter Issue ID to vote (1-3): 3
Vote recorded for Issue 3 by Karan Kapoor. New vote count: 1 on 04/13/2025 at 10:19 PM
Submit - Yes/No: Yes

```

```

Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit

```

```

Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Karan Kapoor
Enter your password: 6778
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 3)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 2)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 1)
Enter Issue ID to vote (1-3): 3
Vote recorded for Issue 3 by Karan Kapoor. New vote count: 2 on 04/13/2025 at 10:20 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Karan Kapoor
Enter your password: 6778
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 1)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 2)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 0)
Enter Proposal ID to vote (1-3): 2
Vote recorded for Proposal 2 by Karan Kapoor. New vote count: 3 on 04/13/2025 at 10:21 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)

```

```

8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Dr. Muskan
Enter your password: 3214
Incorrect username/password or You have not registered yet.
1. Re-enter username and password
2. Register Yourself
Enter your choice (1-2): 2
Welcome to CIVIC PULSE - A Citizen Engagement & Participatory Govt. Portal
-----
Citizen Voice, Govt. Action
-----
Please register to access the portal.
1. Register
2. Exit

```

```

Enter your choice (1-2): 1
Name: Dr. Muskan
Gender: Female
Age: 26
Email: muskan@gmail.com
Aadhar Number: 735373982798
Mobile Number: 9875436782
Enter State: Jharkhand
Enter District: Ranchi
Enter Local Area: Lalpur
Set your password: 2005
Submit - Yes/No: Yes
Notification for Citizen: Registration successful for Dr. Muskan
You have Successfully Registered.
Welcome Dr. Muskan !!
Registration successful. Please try voting again.
Enter your name: Dr. Muskan
Enter your password: 2005
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 3)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 2)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2)
Enter Issue ID to vote (1-3): 1
Vote recorded for Issue 1 by Dr. Muskan. New vote count: 4 on 04/13/2025 at 10:25 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Dr. Muskan
Enter your password: 2005
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 1)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 0)
Enter Proposal ID to vote (1-3): 1
Vote recorded for Proposal 1 by Dr. Muskan. New vote count: 2 on 04/13/2025 at 10:25 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard

```

```

8. Generate Report
10. Exit
Enter your choice (1-10): 2
--- Submit Proposal ---
Enter Description: Small Water ATMs
Enter State: Jharkhand
Enter District: Koderma
Enter Local Area: Jhumri Telaiya
Logging comment internally for Proposal 19
Proposal 19 submitted successfully: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya on 04/13/2025 at 10:29 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Dr. Muskan
Enter your password: 2005
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 2)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 0)
Enter Issue ID to vote (1-4): 4
Vote recorded for Issue 4 by Dr. Muskan. New vote count: 1 on 04/13/2025 at 10:30 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Dr. Muskan
Enter your password: 2005
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 2)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 0)
4. Proposal 4: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya (Votes: 0)
Enter Proposal ID to vote (1-4): 4
Vote recorded for Proposal 4 by Dr. Muskan. New vote count: 1 on 04/13/2025 at 10:31 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit

```



```

Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Vikram Kumar
Enter your password: 3223
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 2)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1)
Enter Issue ID to vote (1-4): 2
Vote recorded for Issue 2 by Vikram Kumar. New vote count: 3 on 04/13/2025 at 10:32 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Vikram Kumar
Enter your password: 3223
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 2)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 0)
4. Proposal 4: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya (Votes: 1)
Enter Proposal ID to vote (1-4): 3
Vote recorded for Proposal 3 by Vikram Kumar. New vote count: 1 on 04/13/2025 at 10:33 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 1
--- Report Issue ---
Enter Description: Frequent Power Cuts
Enter State: West Bengal
Enter District: Burdhaman
Enter Local Area: Durgapur
Logging comment internally for Issue 35
Issue 35 reported successfully: Frequent Power Cuts in West Bengal, Burdhaman, Durgapur on 04/13/2025 at 10:35 PM
Submit - Yes/No: Yes

```

```

Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 2
--- Submit Proposal ---
Enter Description: Free Wi-Fi in Public Places
Enter State: Karnataka
Enter District: Bangalore
Enter Local Area: Yeshwantpur
Logging comment internally for Proposal 26
Proposal 26 submitted successfully: Free Wi-Fi in Public Places in Karnataka, Bangalore, Yeshwantpur on 04/13/2025 at 10:37 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Sudhir Singh
Enter your password: 8907
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Vote on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhan, Durgapur (Votes: 0)
Enter Issue ID to vote (1-5): 5
Vote recorded for Issue 5 by Sudhir Singh. New vote count: 1 on 04/13/2025 at 10:38 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 3
--- Vote on Issue/Proposal ---
Enter your name: Amrit Kumar
Enter your password: 1013
Authentication successful.
Select type to vote on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Vote on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 2)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 1)
4. Proposal 4: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya (Votes: 1)
5. Proposal 5: Free Wi-Fi in Public Places in Karnataka, Bangalore, Yeshwantpur (Votes: 0)
Enter Proposal ID to vote (1-5): 5
Vote recorded for Proposal 5 by Amrit Kumar. New vote count: 1 on 04/13/2025 at 10:38 PM
Submit - Yes/No: Yes
Please select an action:

```

```

Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Comment on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhaman, Durgapur (Votes: 1)
Enter Issue ID to comment (1-5): 3
Enter your comment: Cleaniness is very Essential
Comment added to Issue 39: Cleaniness is very Essential on 04/13/2025 at 10:40 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Comment on Proposal ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 2)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 1)
4. Proposal 4: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya (Votes: 1)
5. Proposal 5: Free Wi-Fi in Public Places in Karnataka, Bangalore, Yeshwantpur (Votes: 1)
Enter Proposal ID to comment (1-5): 4
Enter your comment: Helpful for public
Comment added to Proposal 30: Helpful for public on 04/13/2025 at 10:41 PM

```

```

Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 4
Select type to comment on:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1

```

```

--- Comment on Issue ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhan, Durgapur (Votes: 1)
Enter Issue ID to comment (1-5): 1
Enter your comment: Water should not be waste
Comment added to Issue 39: Water should not be waste on 04/13/2025 at 10:42 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 5
[Authority login check: Enter role (Citizen/Authority): Authority
Select type to review:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Review Issues ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4, Priority: Medium, Status: Under Review)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3, Priority: Medium, Status: Under Review)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2, Priority: Medium, Status: Under Review)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1, Priority: Low, Status: Not Reviewed)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhan, Durgapur (Votes: 1, Priority: Low, Status: Not Reviewed)
Enter Issue ID to review (1-5): 1
Current Priority: Medium (based on 4 votes)
Confirm or adjust priority (Low/Medium/High): High
Issue 1 updated: Water Drainage - Priority: Medium, Status: High on 04/13/2025 at 10:43 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 5
[Authority login check: Enter role (Citizen/Authority): Authority
Select type to review:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Review Issues ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4, Priority: Medium, Status: Under Review)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3, Priority: Medium, Status: Under Review)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2, Priority: Medium, Status: Under Review)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1, Priority: Low, Status: Not Reviewed)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhan, Durgapur (Votes: 1, Priority: Low, Status: Not Reviewed)
Enter Issue ID to review (1-5): 1
Current Priority: Medium (based on 4 votes)
Confirm or adjust priority (Low/Medium/High): High
Issue 1 updated: Water Drainage - Priority: Medium, Status: High on 04/13/2025 at 10:43 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 5
[Authority login check: Enter role (Citizen/Authority): Authority

```

```

Select type to review:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 2
--- Review Proposals ---
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 2, Priority: Medium, Status: Under Review)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3, Priority: Medium, Status: Under Review)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 1, Priority: Low, Status: Not Reviewed)
4. Proposal 4: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya (Votes: 1, Priority: Low, Status: Not Reviewed)
5. Proposal 5: Free Wi-Fi in Public Places in Karnataka, Bangalore, Yeshwantpur (Votes: 1, Priority: Low, Status: Not Reviewed)
Enter Proposal ID to review (1-5): 2
Current Priority: Medium (based on 3 votes)
Confirm or adjust priority (Low/Medium/High): Medium
Proposal 2 updated: Installation of CCTV Cameras - Priority: Medium, Status: Medium on 04/13/2025 at 10:44 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 5
[Authority login check: Enter role (Citizen/Authority): Authority]
Select type to review:
1. Issue Reported
2. Proposal Submission
Enter your choice (1-2): 1
--- Review Issues ---
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4, Priority: Medium, Status: High)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3, Priority: Medium, Status: Under Review)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2, Priority: Medium, Status: Under Review)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1, Priority: Low, Status: Not Reviewed)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhan, Durgapur (Votes: 1, Priority: Low, Status: Not Reviewed)
Enter Issue ID to review (1-5): 4
Current Priority: Low (based on 1 votes)
Confirm or adjust priority (Low/Medium/High): Medium
Issue 4 updated: Water Supply - Priority: Low, Status: Medium on 04/13/2025 at 10:45 PM
Submit - Yes/No: Yes
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 7
--- Dashboard ---
Dashboard Updated on 04/13/2025 at 10:47 PM
-----
Issues Reported:
1. Issue 1: Water Drainage in Karnataka, Bangalore, Kogilu Cross (Votes: 4, Priority: Medium, Status: High)
2. Issue 2: Poor road conditions in Goa, Panji, GR Road (Votes: 3, Priority: Medium, Status: Under Review)
3. Issue 3: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali (Votes: 2, Priority: Medium, Status: Under Review)
4. Issue 4: Water Supply in Jharkhand, Ranchi, Kanta Toli (Votes: 1, Priority: Low, Status: Medium)
5. Issue 5: Frequent Power Cuts in West Bengal, Burdhan, Durgapur (Votes: 1, Priority: Low, Status: Not Reviewed)
-----
Proposals Submitted:
1. Proposal 1: Build a Water tank in Bihar, Gaya, PLA Road (Votes: 2, Priority: Medium, Status: Under Review)
2. Proposal 2: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle (Votes: 3, Priority: Medium, Status: Medium)
3. Proposal 3: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh (Votes: 1, Priority: Low, Status: Not Reviewed)
4. Proposal 4: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya (Votes: 1, Priority: Low, Status: Not Reviewed)
5. Proposal 5: Free Wi-Fi in Public Places in Karnataka, Bangalore, Yeshwantpur (Votes: 1, Priority: Low, Status: Not Reviewed)
-----
Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit

```

```

Please select an action:
1. Report Issue (Citizen)
2. Submit Proposal (Citizen)
3. Vote on Issue/Proposal (Citizen)
4. Comment on Issue/Proposal (Citizen)
5. Review and Prioritize (Authority)
7. View Dashboard
8. Generate Report
10. Exit
Enter your choice (1-10): 8
=== CIVIC PULSE - Comprehensive Report ===
Report Generated on: 04/13/2025 at 10:49 PM
-----
** Registration Details **
Name: Dr. Muskan
Gender: Female
Age: 26
Email: muskan@gmail.com
Aadhar Number: 735373982798
Mobile Number: 9875436782
Address: Jharkhand, Ranchi, Lalpur
Registration Date: 01/01/1970 at 05:30 AM
-----
** Issues Reported **
1. Issue: Water Drainage in Karnataka, Bangalore, Kogilu Cross
   Reported by: Amrit Kumar
   Reported on: 04/13/2025 at 09:51 PM
   Vote Count: 4
   Comments: Cleaniness is very Essential by Dr. Muskan on 04/13/2025 at 10:40 PM, Water should not be waste by Dr. Muskan on 04/13/2025 at 10:42 PM
   Priority: Medium
   Status: High
   -----
2. Issue: Poor road conditions in Goa, Panji, GR Road
   Reported by: Vikram Kumar
   Reported on: 04/13/2025 at 10:03 PM
   Vote Count: 3
   Comments: Cleaniness is very Essential by Dr. Muskan on 04/13/2025 at 10:40 PM, Water should not be waste by Dr. Muskan on 04/13/2025 at 10:42 PM
   Priority: Medium
   Status: Under Review
   -----
3. Issue: Irregular Garbage Collection in Uttar Pradesh, Lucknow, Chatori Gali
   Reported by: Sudhir Singh
   Reported on: 04/13/2025 at 10:15 PM
   Vote Count: 2
   Comments: Cleaniness is very Essential by Dr. Muskan on 04/13/2025 at 10:40 PM, Water should not be waste by Dr. Muskan on 04/13/2025 at 10:42 PM
   -----
4. Issue: Water Supply in Jharkhand, Ranchi, Kanta Toli
   Reported by: Dr. Muskan
   Reported on: 04/13/2025 at 10:26 PM
   Vote Count: 1
   Comments: Cleaniness is very Essential by Dr. Muskan on 04/13/2025 at 10:40 PM, Water should not be waste by Dr. Muskan on 04/13/2025 at 10:42 PM
   Priority: Low
   Status: Medium
   -----
5. Issue: Frequent Power Cuts in West Bengal, Burdhan, Durgapur
   Reported by: Dr. Muskan
   Reported on: 04/13/2025 at 10:35 PM
   Vote Count: 1
   Comments: Cleaniness is very Essential by Dr. Muskan on 04/13/2025 at 10:40 PM, Water should not be waste by Dr. Muskan on 04/13/2025 at 10:42 PM
   Priority: Low
   Status: Not Reviewed
   -----
** Proposals Submitted **
1. Proposal: Build a Water tank in Bihar, Gaya, PLA Road
   Reported by: Amrit Kumar
   Reported on: 04/13/2025 at 09:52 PM
   Vote Count: 2
   Comments: Helpful for public by Dr. Muskan on 04/13/2025 at 10:41 PM
   Priority: Medium
   Status: Under Review
   -----
2. Proposal: Installation of CCTV Cameras in Delhi, New Delhi, Green Circle
   Reported by: Vikram Kumar
   Reported on: 04/13/2025 at 10:05 PM
   Vote Count: 3
   Comments: Helpful for public by Dr. Muskan on 04/13/2025 at 10:41 PM
   Priority: Medium
   Status: Medium
   -----
3. Proposal: Build a Local Community Park in Uttar Pradesh, Lucknow, Charbagh
   Reported by: Sudhir Singh
   Reported on: 04/13/2025 at 10:12 PM
   Vote Count: 1
   Comments: Helpful for public by Dr. Muskan on 04/13/2025 at 10:41 PM
   Priority: Low
   Status: Not Reviewed
   -----
4. Proposal: Small Water ATMs in Jharkhand, Koderma, Jhumri Telaiya
   Reported by: Dr. Muskan
   Reported on: 04/13/2025 at 10:29 PM
   Vote Count: 1
   Comments: Helpful for public by Dr. Muskan on 04/13/2025 at 10:41 PM

```

## 10. Conclusion

The "Civic Pulse" project stands as a remarkable achievement in the realm of participatory governance, seamlessly blending cutting-edge technology with the principles of community engagement and administrative efficiency. Through a meticulously crafted design process, this initiative has evolved into a robust and scalable platform that empowers Citizens and Authorities alike, fostering a transparent and inclusive civic ecosystem. The integration of advanced features—such as the enhanced User class with a password attribute, the refined Citizen constructor, and the innovative registeredUsers list with the authenticate method—underscores a steadfast commitment to security and user verification, ensuring that every interaction is both trustworthy and reliable.

The thoughtfully designed Class Diagram, reoriented to a top-to-bottom flow, enhances readability and maintainability, while the updated Sequence Diagrams and Use Case Diagrams provide a clear and actionable blueprint for implementation. From the seamless registration process, complete with a dedicated "Set Password" step, to the democratic empowerment of voting with robust authentication, every aspect of the system reflects a harmonious balance of functionality and foresight. The inclusion of diverse use cases—ranging from reporting issues and submitting proposals to reviewing and prioritizing them—demonstrates the project's versatility, catering to the needs of a dynamic civic landscape.

As of April 13, 2025, "Civic Pulse" emerges as a beacon of innovation, poised to transform how communities interact with governance. Its foundation in abstraction, inheritance, and interface-driven design ensures extensibility, paving the way for future enhancements and integrations, potentially with xAI's API services to further amplify its capabilities. With rigorous validation mechanisms, comprehensive transaction logging, and a user-friendly CLI interface, the project not only meets its current objectives but also sets a high standard for future civic technology initiatives. "Civic Pulse" is more than a system—it is a catalyst for empowerment, collaboration, and progress, leaving a lasting legacy in the journey toward a more engaged and responsive society.



## 11. References

1. **Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide* (2nd ed.). Addison-Wesley Professional.**
  - A foundational resource on UML, providing guidelines for creating Class Diagrams, Sequence Diagrams, and Use Case Diagrams, which were central to designing the "Civic Pulse" system architecture.
2. **Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.**
  - Offers insights into software design principles, including abstraction, inheritance, and exception handling, which were applied to ensure the robustness and maintainability of the "Civic Pulse" project.
3. **Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.**
  - Provides patterns for system design, such as the use of interfaces and composition (e.g., User with Address), which influenced the extensible structure of "Civic Pulse."
4. **United Nations. (2018). *E-Government Survey 2018: Gearing E-Government to Support Transformation Towards Sustainable and Resilient Societies*. United Nations Publications.**
  - Highlights the role of e-governance platforms in civic engagement, offering a real-world context for the "Civic Pulse" project's goal of enhancing participatory governance.
5. **Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.**
  - A classic reference on design patterns, supporting the use of inheritance and polymorphism in the "Civic Pulse" Class Diagram (e.g., User to Citizen and Authority).
6. **ISO/IEC 25010:2011. *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models*. International Organization for Standardization.**
  - Provides standards for software quality attributes (e.g., security, usability), which guided the inclusion of authentication and validation in "Civic Pulse."



## **PHASE 2**

## **CIVIC PULSE**

## **Operations Required**

### **1. Issue Processing and Vendor Assignment (Phase 2)**

- **Description:**

Facilitate the review, assignment, and processing of reported issues by authorities, ensuring compliance with resolution workflows, vendor capacity constraints, and deadline enforcement for quotations.

- **Specific Requirements:**

- **Class:** Assignment with:

- private String assignmentId: Unique identifier for the assignment.
- private String issueId: Links to the issue being assigned.
- private String vendorId: Links to the assigned vendor (optional for auto-assignment).
- private Date submissionDate: Timestamp of the assignment request.
- private Date quotationDeadline: Deadline for quotation submission (48 hours from submission).
- private String status: Current status (e.g., "Quotations Requested", "Vendor Selected", "Work in Progress").
- private List<Vendor> potentialVendors: List of vendors eligible based on expertise.
- public void setStatus(String status): Updates assignment status with validation.
- public void cancelAssignment(): Cancels assignment if status allows (e.g., before "Work in Progress").
- public void addPotentialVendor(Vendor vendor): Adds vendor to potential list for auto-assignment.
- public Vendor autoAssignVendor(): Automatically assigns vendor based on expertise and workload.
- protected boolean validateAssignmentState(): Ensures issue is "Open" and assignment status is valid for transitions.
- protected boolean checkDeadline(): Verifies if quotation deadline has passed.

- **Inner Class:** AssignmentDetails within Assignment with:
  - private String assignmentType: Type of assignment (Manual, Auto).
  - private Date lastStatusChange: Timestamp of the last status update.
  - public String getAssignmentType(): Retrieves assignment type for logging.
- **Sub Classes (Multilevel Inheritance):**
  - PendingAssignment extends Assignment with:
    - private Date assignmentStartDate: Timestamp of assignment initiation.
    - private boolean isWaiting: Indicates if assignment is on waiting list.
    - public void processAssignment(): Initiates vendor assignment process (auto or manual).
    - public void processAssignment(String vendorId): Overloaded for manual vendor selection.
    - public void moveToWaitingList(): Adds assignment to waiting list if no vendors are available.
  - AssignedAssignment extends PendingAssignment with:
    - private String confirmedVendorId: Vendor confirmed after quotation selection.
    - private Date vendorConfirmationDate: Timestamp of vendor confirmation.
    - protected void finalizeAssignment(): Completes assignment by updating status to "Vendor Selected".
    - public void startWork(): Transitions status to "Work in Progress".
- **Constructors:**
  - Assignment(String issueld): Initializes with issue ID, auto-generates assignmentId.
  - Assignment(String issueld, String vendorId): Includes manual vendor selection.
  - Assignment(String issueld, List<Vendor> potentialVendors): Includes list for auto-assignment.

- **Collections:**
  - `ArrayList<Assignment>` `assignments` in `CivicPulseSystem`: Stores all assignments dynamically.
  - `ArrayList<Assignment>` `waitingList` in `CivicPulseSystem`: Manages assignments waiting for vendor availability.
- **Method with Object Reference:**
  - `public void processAssignment(Assignment assignment)` in `CivicPulseSystem`: Processes assignment with priority-based vendor selection.
  - `public void reassignFromWaitingList(Vendor vendor)`: Reassigns waiting assignments when vendor capacity opens.
- **Exceptions:**
  - `IssueNotFoundException` (user-defined): Thrown if `issueld` is invalid.
  - `VendorUnavailableException` (user-defined): Thrown if no vendors are available or capacity is exceeded.
  - `QuotationDeadlineExceededException` (user-defined): Thrown if quotations are not received within 48 hours.
  - `InvalidStateException` (user-defined): Thrown if assignment state does not allow the operation (e.g., canceling after work starts).
- **New Class:** `AssignmentRules` with:
  - `private int maxAssignmentsPerVendor`: Maximum concurrent assignments per vendor (e.g., 5).
  - `private int quotationDeadlineHours`: Quotation submission deadline (48 hours).
  - `private int maxPotentialVendors`: Maximum vendors to consider for auto-assignment (e.g., 3).
  - `public boolean canAssignVendor(Vendor vendor, Issue issue)`: Validates vendor assignment based on expertise and workload.
  - `public boolean isWithinDeadline(Assignment assignment)`: Checks if assignment is within quotation deadline.
- **New Class:** `VendorSelectionCriteria` with:

- `private String expertise`: Required expertise for the issue.
- `private int maxWorkload`: Maximum workload threshold for selection.
- `public List<Vendor> filterVendors(List<Vendor> vendors, Issue issue)`:  
Filters vendors based on criteria.

- **Justification:**

- **Private**: `assignmentId`, `issueId`, `vendorId`, `submissionDate`, `quotationDeadline`, `status`, `potentialVendors` ensure encapsulation, securing assignment data and preventing unauthorized modifications (checklist: Class with access specifiers).
- **Protected**: `validateAssignmentState()`, `checkDeadline()` allow inheritance flexibility for future assignment types, enabling subclasses to customize validation logic (checklist: Protected).
- **Public**: `setStatus()`, `cancelAssignment()`, `autoAssignVendor()` are accessible for system operations, justified for managing assignment workflows and ensuring flexibility in vendor selection (checklist: Class).
- **Inner Class**: `AssignmentDetails` adds modularity by encapsulating metadata like assignment type, supporting structured design and logging requirements (checklist: Inner Class).
- **Multilevel Inheritance**: `Assignment` → `PendingAssignment` → `AssignedAssignment` structures the state transition process, ensuring clear separation of pending and assigned states, meeting checklist requirements (checklist: Inheritance Multilevel).
- **Collections**: `ArrayList<Assignment>` for assignments and `waitingList` supports scalability, allowing dynamic management of assignments and waiting scenarios, while `List<Vendor>` in `potentialVendors` enables efficient vendor selection (checklist: Collections 1).
- **Exceptions**: `IssueNotFoundException`, `VendorUnavailableException`, `QuotationDeadlineExceededException`, `InvalidStateException` enhance error handling by addressing edge cases like invalid IDs, unavailable vendors, late quotations, and invalid state transitions, ensuring robust system behavior (checklist: Exceptions, User-Defined Exceptions).
- **Sequence Integration**: `processAssignment()` initiates the assignment process, calling `autoAssignVendor()` or manual assignment, followed by

moveToWaitingList() if needed, ensuring seamless workflow integration  
(checklist: Consistency with Method in Java Code).

---

## 2. Vendor Management (Phase 2)

- **Description:**

Manage vendors with hierarchical inheritance, supporting capacity tracking, expertise-based assignment, and real-time workload updates for efficient issue resolution.

- **Specific Requirements:**

- **Interface (Abstraction):** Manageable with:
  - public void updateResource(): Defines a contract for managing vendor resources (e.g., workload).
  - public boolean canAcceptIssue(Issue issue): Checks if vendor can accept an issue based on expertise and capacity.
- **Class:** Vendor implements Manageable with:
  - private String vendorId: Unique identifier for the vendor.
  - public String name: Vendor name, publicly visible for selection.
  - private String contactInfo: Contact details (e.g., email, phone), sensitive data.
  - private String[] expertiseAreas: Areas of expertise (e.g., "infrastructure", "sanitation").
  - private int capacity: Maximum concurrent issues the vendor can handle (e.g., 5).
  - private int currentWorkload: Current number of assigned issues.
  - private List<Assignment> activeAssignments: Tracks currently assigned issues.
  - private List<Quotation> submittedQuotations: Tracks quotations submitted by the vendor.
  - private double performanceRating: Vendor's average rating (1-5 scale).
  - public boolean assignIssue(Issue issue): Assigns issue if capacity and expertise match.

- `public void addToAssignments(Assignment assignment):` Adds assignment to active list.
- `public void submitQuotation(Issue issue, double price, int estimatedDays):` Submits a quotation for an issue.
- `public void updateWorkload(int delta):` Updates workload (e.g., +1 for new assignment, -1 for completion).
- `public void updatePerformanceRating(double rating):` Updates rating based on feedback.
- `protected void updateResource():` Implements interface method to manage workload.
- `protected boolean validateExpertise(Issue issue):` Validates if vendor's expertise matches issue category.
- **Inner Class:** `VendorMetrics` within `Vendor` with:
  - `private int totalAssignments:` Total number of assignments handled historically.
  - `private double averageCompletionTime:` Average days to complete assignments.
  - `public double getAverageCompletionTime():` Retrieves metric for reporting.
- **Sub Classes (Hierarchical Inheritance):**
  - `InfrastructureVendor` extends `Vendor` with:
    - `private String equipmentType:` Infrastructure-specific data (e.g., "Heavy Machinery").
    - `private int maxEquipmentUsage:` Maximum equipment usage limit (e.g., 3 projects).
    - `public boolean assignIssue(Issue issue):` Overrides to include equipment availability check.
    - `public void checkEquipmentAvailability():` Ensures equipment is available for assignment.
  - `SanitationVendor` extends `Vendor` with:
    - `private String cleaningMethod:` Sanitation-specific data (e.g., "Chemical Cleaning").
    - `private int cleaningCrewSize:` Number of crew members available.

- `public boolean assignIssue(Issue issue)`: Overrides to include crew size check.
- `public void allocateCrew(int crewSize)`: Allocates crew for the assignment.
- **Constructors:**
  - `Vendor(String vendorId, String name, String contactInfo, String[] expertiseAreas, int capacity)`: Full initialization with all details.
  - `Vendor(String vendorId, String name, String contactInfo)`: Defaults capacity to 5 and expertise to ["General"].
  - `Vendor(String vendorId, String name)`: Minimal initialization, defaults contactInfo to "N/A".
- **Collections:**
  - `HashMap<String, Vendor> vendors` in `CivicPulseSystem`: Enables fast lookups by vendor ID.
  - `List<Assignment> activeAssignments`: Tracks assignments per vendor.
  - `List<Quotation> submittedQuotations`: Tracks quotations per vendor.
- **Method with Object Reference:**
  - `public List<Vendor> searchVendorsByExpertise(String expertise)` in `CivicPulseSystem`: Searches vendors by expertise.
  - `public List<Vendor> searchVendorsByName(String name)`: Searches vendors by name.
- **Exceptions:**
  - `VendorCapacityFullException` (user-defined): Thrown when vendor capacity is exceeded.
  - `ExpertiseMismatchException` (user-defined): Thrown when vendor expertise does not match issue category.
- **New Class:** `VendorPerformance` with:
  - `private double minRatingThreshold`: Minimum rating for eligibility (e.g., 3.0).
  - `private int maxWorkloadThreshold`: Maximum workload before performance review (e.g., 4).
  - `public boolean isEligibleForAssignment(Vendor vendor)`: Checks if vendor meets performance criteria.



- `public void calculatePerformanceMetrics(Vendor vendor)`: Updates performance metrics based on completed assignments.

- **Justification:**

- **Private:** `vendorId`, `contactInfo`, `expertiseAreas`, `capacity`, `currentWorkload`, `activeAssignments`, `submittedQuotations`, `performanceRating` ensure encapsulation, protecting sensitive vendor data and internal state from unauthorized access (checklist: Class with access specifiers).
- **Protected:** `updateResource()`, `validateExpertise()` allow inheritance flexibility, enabling subclasses like `InfrastructureVendor` to customize resource management and expertise validation logic (checklist: Protected).
- **Public:** `name`, `assignIssue()`, `submitQuotation()`, `updateWorkload()` are accessible for system operations, justified for enabling issue assignment, quotation submission, and workload management (checklist: Class).
- **Inner Class:** `VendorMetrics` adds modularity by encapsulating performance metrics, supporting structured design and facilitating reporting requirements (checklist: Inner Class).
- **Interface:** `Manageable` provides abstraction by defining a clear contract for resource management, ensuring all vendors adhere to a standard workload update process (checklist: Abstraction through Interface).
- **Hierarchical Inheritance:** `Vendor` → `InfrastructureVendor`/`SanitationVendor` enables specialized vendor types with tailored assignment logic, meeting checklist requirements for hierarchical structure (checklist: Inheritance Hierarchical).
- **Collections:** `HashMap<String, Vendor>` in `CivicPulseSystem` supports fast lookups for vendor selection, while `List<Assignment>` and `List<Quotation>` per vendor enable dynamic tracking of assignments and quotations, ensuring scalability (checklist: Collections 1, 2).
- **Exceptions:** `VendorCapacityFullException` and `ExpertiseMismatchException` enhance error handling by addressing capacity and expertise mismatches, ensuring only qualified vendors are assigned to issues (checklist: Exceptions, User-Defined Exceptions).
- **Sequence Integration:** `assignIssue()` is called during the assignment process, followed by `updateWorkload()` to reflect changes in vendor capacity, ensuring

seamless integration with the workflow (checklist: Consistency with Method in Java Code).

---

### 3. Quotation Management (Phase 2)

- **Description:**

Enable vendors to submit, revise, and track quotations for assigned issues, with authorities selecting the most cost-effective option using a tiebreaker mechanism to ensure optimal resolution costs.

- **Specific Requirements:**

- **Class:** Quotation with:

- private String quotationId: Unique identifier for the quotation.
    - private String issueld: Links to the issue being quoted.
    - private String vendorId: Links to the submitting vendor.
    - private double price: Quotation price (must be positive, e.g., > 0.01).
    - private int estimatedDays: Estimated completion time (must be positive, e.g., >= 1).
    - private String details: Additional details (e.g., "Materials: Concrete, Labor: 3 workers").
    - private Date submissionDate: Timestamp of quotation submission.
    - private Date deadline: Submission deadline (48 hours from assignment).
    - private String status: Current status (e.g., "Pending", "Selected", "Rejected").
    - private double vendorRatingAtSubmission: Vendor's rating at the time of submission (for tiebreaker).
    - public void reviseQuotation(double newPrice, int newDays): Updates price and days before selection.
    - public void reviseQuotation(double newPrice, int newDays, String newDetails): Overloaded with updated details.
    - public void setStatus(String status): Updates status with validation (e.g., cannot change "Selected" to "Pending").

- `public double calculateAdjustedPrice():` Adjusts price based on vendor rating (e.g., higher rating reduces perceived cost).
- `protected boolean validateQuotationInputs():` Validates price, estimated days, and deadline adherence.
- `protected boolean isWithinDeadline():` Checks if submission or revision is within the 48-hour deadline.
- **Inner Class:** `QuotationDetails` within `Quotation` with:
  - `private String submissionMethod:` Method of submission (e.g., "Manual", "Automated").
  - `private int revisionCount:` Number of revisions made to the quotation.
  - `public int getRevisionCount():` Retrieves revision count for auditing.
- **Enum:** `QuotationStatus` with `PENDING`, `SELECTED`, `REJECTED`, `EXPIRED`.
- **Constructors:**
  - `Quotation(String issuelId, String vendorId, double price, int estimatedDays):` Basic initialization with mandatory fields.
  - `Quotation(String issuelId, String vendorId, double price, int estimatedDays, String details):` Includes additional details.
  - `Quotation(String issuelId, String vendorId, double price, int estimatedDays, Date deadline):` Includes custom deadline.
- **Collections:**
  - `ArrayList<Quotation> quotations` in `Issue`: Stores all quotations for an issue dynamically.
  - `List<Quotation> submittedQuotations` in `Vendor`: Tracks quotations submitted by a vendor.
- **Method with Object Reference:**
  - `public Quotation findLowestPriceQuotation(List<Quotation> quotations)` in `CivicPulseSystem`: Identifies the lowest price quotation, using vendor rating as a tiebreaker if prices are equal.
  - `public void notifyVendor(Quotation quotation, String message):` Notifies vendor of selection or rejection.
- **Exceptions:**
  - `InvalidQuotationException` (user-defined): Thrown for invalid price, days, or late submission.

- **DeadlineExceededException** (user-defined): Thrown if quotation submission or revision exceeds the 48-hour deadline.
- **InvalidStateException** (user-defined): Thrown if attempting to revise a "Selected" or "Rejected" quotation.
- **New Class:** QuotationRules with:
  - private double minPrice: Minimum allowable price (e.g., 0.01).
  - private int minDays: Minimum allowable days (e.g., 1).
  - private int maxDays: Maximum allowable days (e.g., 30).
  - private double ratingWeight: Weight of vendor rating in tiebreaker (e.g., 0.1).
  - public boolean isValidQuotation(Quotation quotation): Validates quotation against price, days, and deadline rules.
  - public double calculateTiebreakerScore(Quotation quotation): Calculates score for tiebreaker (price adjusted by rating).
- **New Class:** QuotationAudit with:
  - private List<String> auditLogs: Logs all quotation actions (e.g., submission, revision, selection).
  - private Date lastAudit: Timestamp of last audit.
  - public void logAction(String action, Quotation quotation): Logs quotation-related actions for transparency.
  - public List<String> getAuditLogs(): Retrieves logs for review.
- **Justification:**
  - **Private:** quotationId, issuedId, vendorId, price, estimatedDays, details, submissionDate, deadline, status, vendorRatingAtSubmission ensure encapsulation, protecting sensitive quotation data from unauthorized access (checklist: Class with access specifiers).
  - **Protected:** validateQuotationInputs(), isWithinDeadline() allow inheritance flexibility for future quotation types, enabling subclasses to customize validation logic (checklist: Protected).
  - **Public:** reviseQuotation(), setStatus(), calculateAdjustedPrice() are accessible for system operations, justified for enabling quotation revisions, status updates, and optimized selection (checklist: Class).

- **Inner Class:** QuotationDetails adds modularity by encapsulating metadata like revision count, supporting auditing and transparency requirements (checklist: Inner Class).
  - **Enum:** QuotationStatus ensures consistent status values across the system, reducing errors in status management (checklist: Class).
  - **Collections:** ArrayList<Quotation> in Issue and List<Quotation> in Vendor support scalability, allowing dynamic management of quotations per issue and per vendor (checklist: Collections 1).
  - **Exceptions:** InvalidQuotationException, DeadlineExceededException, InvalidStateException enhance error handling by addressing invalid inputs, late submissions, and invalid state transitions, ensuring robust quotation management (checklist: Exceptions, User-Defined Exceptions).
  - **Sequence Integration:** findLowestPriceQuotation() is called during the quotation selection process, followed by notifyVendor() to inform the selected vendor, ensuring seamless integration with the workflow (checklist: Consistency with Method in Java Code).
- 

## 4. Work Progress Notification (Phase 2)

- **Description:**

Deliver prioritized notifications to authorities and citizens about work progress, start, and completion events, with detailed logging and mock delivery mechanisms to simulate real-world communication.

- **Specific Requirements:**

- **Class:** Notification with:
  - private String notificationId: Unique identifier for the notification.
  - private String message: Notification content (e.g., "Work on ISSUE001 started").
  - private NotificationType type: Categorizes notification (INFO, PROGRESS, COMPLETION).
  - private int priority: Determines urgency (1-5, where 1 is highest).
  - private String senderId: ID of the sender (e.g., system-generated).

- `public Date timestamp`: Timestamp of notification creation.
- `public void send()`: Simulates delivery by logging to console (e.g., `System.out.println(message)`).
- `public void resend()`: Attempts to resend if delivery fails (simulated).
- `protected boolean validateNotification()`: Ensures message is non-empty and priority is valid.
- **Inner Class**: `NotificationDetails` within `Notification` with:
  - `private String recipientId`: Links to the recipient (`citizenId` or `authorityId`).
  - `private String deliveryStatus`: Status of delivery (Sent, Pending, Failed).
  - `private int retryCount`: Number of delivery retries (max 3).
  - `public String getDeliveryStatus()`: Retrieves status for tracking.
  - `public void incrementRetryCount()`: Increments retry count on failure.
- **Enum**: `NotificationType` with values `INFO`, `PROGRESS`, `COMPLETION`, `ERROR`.
- **Constructors**:
  - `Notification(String message, NotificationType type, int priority)`: Basic initialization for system-generated notifications.
  - `Notification(String message, String recipientId, NotificationType type, int priority)`: Includes recipient for targeted notifications.
  - `Notification(String message, String senderId, String recipientId, NotificationType type, int priority)`: Full initialization with sender details.
- **Method without Object Reference**:
  - `public static String generateId()`: Generates a unique notification ID (e.g., "NOTIF001").
  - `public static boolean validatePriority(int priority)`: Ensures priority is between 1 and 5.
- **Collections**:
  - `LinkedList<Notification> notifications` in `CivicPulseSystem`: Stores notifications in order of priority and timestamp for delivery.
  - `List<Notification> notificationHistory` in `Citizen` and `Authority`: Tracks received notifications per user.
- **Method with Object Reference**:

- `public void sendNotification(Notification notification)` in `CivicPulseSystem`:  
Sends the notification and updates its status.
  - `public List<Notification> getNotificationsByRecipient(String recipientId)`:  
Retrieves notification history for a recipient.
- **Exceptions:**
  - `IllegalArgumentException`: Thrown for invalid messages (e.g., empty) or priorities (e.g.,  $< 1$  or  $> 5$ ).
  - `DeliveryFailedException` (user-defined): Thrown if delivery fails after maximum retries (simulated).
- **New Class:** `NotificationQueue` with:
  - `private PriorityQueue<Notification> queue`: Priority queue for notifications based on priority.
  - `private int maxQueueSize`: Maximum queue size (e.g., 1000).
  - `public void addNotification(Notification notification)`: Adds notification to queue.
  - `public Notification processNextNotification()`: Processes the highest-priority notification.
  - `public boolean isQueueFull()`: Checks if queue has reached capacity.
- **New Class:** `NotificationLogger` with:
  - `private List<String> logs`: Logs all notification actions (e.g., sent, failed).
  - `private Date lastLog`: Timestamp of the last log entry.
  - `public void logNotificationAction(String action, Notification notification)`:  
Logs action with timestamp.
  - `public List<String> getLogs()`: Retrieves logs for auditing.
- **Justification:**
  - **Private:** `notificationId`, `message`, `type`, `priority`, `senderId` ensure encapsulation, protecting sensitive notification data from unauthorized access (checklist: Class with access specifiers).
  - **Protected:** `validateNotification()` allows inheritance flexibility for future notification types, enabling subclasses to customize validation logic (checklist: Protected).
  - **Public:** `timestamp`, `send()`, `resend()` are accessible for system operations, justified for enabling notification delivery, retry mechanisms, and tracking (checklist: Class).

- **Inner Class:** NotificationDetails adds modularity by encapsulating delivery details, supporting structured design and facilitating tracking of delivery status (checklist: Inner Class).
  - **Enum:** NotificationType ensures consistent categorization of notifications, reducing errors in type management (checklist: Class).
  - **Collections:** LinkedList<Notification> in CivicPulseSystem and List<Notification> in Citizen/Authority support scalability, ensuring ordered delivery and historical tracking of notifications (checklist: Collections 3).
  - **Collections:** PriorityQueue<Notification> in NotificationQueue ensures high-priority notifications are processed first, optimizing communication (checklist: Collections 2).
  - **Exceptions:** IllegalArgumentException and DeliveryFailedException enhance error handling by addressing invalid inputs and failed deliveries, ensuring reliable notification processing (checklist: Exceptions, User-Defined Exceptions).
  - **Sequence Integration:** sendNotification() is called after work progress updates, such as starting work or completing an issue, ensuring stakeholders are informed in real-time (checklist: Consistency with Method in Java Code).
- 

## 5. Role-Based Access for Authority and Administrator (Phase 2)

- **Description:**

Define specific roles for authorities and administrators with permissions to manage issue resolution workflows, ensuring secure and controlled access to system functionalities.
- **Specific Requirements:**
  - **Interface:** Reportable with:
    - public void generateReport(): Defines a contract for generating reports.
    - public String getReportSummary(): Retrieves a summary of the report.
  - **Interface:** Manageable with:
    - public void updateResource(): Defines a contract for managing system resources.
    - public void auditResources(): Audits resource usage for compliance.



- **Class:** Authority extends User implements Manageable with:
  - private String authorityId: Unique identifier for the authority.
  - private List<Assignment> managedAssignments: Tracks assignments overseen by the authority.
  - private List<Grievance> assignedGrievances: Tracks grievances assigned for resolution.
  - public void assignVendor(Issue issue, Vendor vendor): Assigns a vendor to an issue manually.
  - public void assignVendor(Issue issue): Overloaded for automatic assignment.
  - public void selectQuotation(String issueId, String quotationId): Selects a quotation for an issue.
  - public void updateWorkProgress(String assignmentId, String progress): Updates progress for an assignment (e.g., "50% complete").
  - public void reviewGrievance(String grievanceId): Reviews a grievance before resolution.
  - public void updateResource(): Implements interface method to manage workload.
  - public void auditResources(): Audits resource usage (e.g., vendor assignments).
  - protected boolean validateAssignmentPermissions(Issue issue): Ensures authority has permission to assign the issue.
- **Class:** Administrator extends Authority implements Reportable with:
  - private String adminCode: Unique identifier for the administrator.
  - private List<Vendor> managedVendors: Tracks vendors managed by the admin.
  - private List<User> managedUsers: Tracks users managed by the admin.
  - public void manageUsers(): Adds, updates, or removes user accounts.
  - public void manageVendors(): Adds, updates, or removes vendors from the system.
  - public void manageVendors(String vendorId): Overloaded for specific vendor management.

- `public void generateReport():` Generates system-wide reports (e.g., issue stats, vendor performance).
- `public String getReportSummary():` Retrieves a summary of the latest report.
- `public void resolveGrievance(String grievanceId, String resolution):` Resolves a grievance with notes.
- `public void auditSystem():` Performs a full system audit (e.g., user activity, vendor performance).
- **Inner Class:** `AuthorityLog` within `Authority` with:
  - `private List<String> actionLogs:` Logs all actions performed by the authority (e.g., "Assigned vendor to ISSUE001").
  - `private Date lastAction:` Timestamp of the last action.
  - `public void logAction(String action):` Logs an action with timestamp.
  - `public List<String> getActionLogs():` Retrieves logs for auditing.
- **Constructors:**
  - `Authority(String name, String authorityId):` Initializes authority with basic details.
  - `Authority(String name, String authorityId, List<Assignment> managedAssignments):` Includes managed assignments.
  - `Administrator(String name, String adminCode):` Initializes admin with basic details.
  - `Administrator(String name, String adminCode, List<Vendor> managedVendors):` Includes managed vendors.
- **Method without Object Reference:**
  - `public static boolean validateRole(String role):` Validates if a role is allowed (e.g., "Authority", "Administrator").
  - `public static String generateAdminCode():` Generates a unique admin code (e.g., "ADMIN001").
- **Collections:**
  - `List<Assignment> managedAssignments:` Tracks assignments per authority.
  - `List<Grievance> assignedGrievances:` Tracks grievances per authority.
  - `List<Vendor> managedVendors:` Tracks vendors per administrator.

- List<User> managedUsers: Tracks users per administrator.
- **Exceptions:**
  - PermissionDeniedException (user-defined): Thrown if an authority or admin lacks permission for an action.
  - InvalidGrievanceException (user-defined): Thrown if a grievance cannot be resolved due to invalid state.
- **New Class:** RolePermissions with:
  - private String role: Role name (e.g., "Authority", "Administrator").
  - private List<String> permissions: List of allowed actions (e.g., "AssignVendor", "ResolveGrievance").
  - private Map<String, Boolean> permissionCache: Cache for quick permission lookups.
  - public boolean hasPermission(String action): Checks if the role has permission for the action.
  - public void addPermission(String action): Adds a new permission to the role.
  - public void removePermission(String action): Removes a permission from the role.
- **New Class:** AuditRecord with:
  - private String auditId: Unique identifier for the audit record.
  - private String auditedEntity: Entity being audited (e.g., "Vendor", "Assignment").
  - private Date auditDate: Timestamp of the audit.
  - private List<String> findings: List of audit findings (e.g., "Vendor over capacity").
  - public void addFinding(String finding): Adds a finding to the record.
  - public List<String> getFindings(): Retrieves findings for review.
- **Justification:**
  - **Private:** authorityId, adminCode, managedAssignments, assignedGrievances, managedVendors, managedUsers, role, permissions, permissionCache ensure encapsulation, securing sensitive role data and preventing unauthorized access (checklist: Class with access specifiers).

- **Protected:** `validateAssignmentPermissions()` allows inheritance flexibility, enabling subclasses to customize permission validation logic (checklist: Protected).
- **Public:** `assignVendor()`, `selectQuotation()`, `generateReport()`, `resolveGrievance()` are accessible for role-specific operations, justified for enabling authorities and administrators to manage workflows (checklist: Class).
- **Inner Class:** `AuthorityLog` adds modularity by encapsulating action logs, supporting structured design and facilitating auditing requirements (checklist: Inner Class).
- **Interfaces:** `Manageable` and `Reportable` provide abstraction, simulating multiple inheritance by defining clear contracts for resource management and reporting, ensuring all roles adhere to standard behavior (checklist: Abstraction through Interface, Multiple Inheritance).
- **Multilevel Inheritance:** `User` → `Authority` → `Administrator` structures role hierarchy, ensuring shared functionality (e.g., `updateResource()`) while allowing specialized behavior (e.g., `generateReport()`), meeting checklist requirements (checklist: Inheritance Multilevel).
- **Collections:** `List<Assignment>`, `List<Grievance>`, `List<Vendor>`, `List<User>`, `List<String>` in `RolePermissions`, and `Map<String, Boolean>` in `permissionCache` support scalability, managing dynamic data for assignments, grievances, vendors, users, permissions, and caching (checklist: Collections 1, 2).
- **Exceptions:** `PermissionDeniedException` and `InvalidGrievanceException` enhance error handling by addressing unauthorized actions and invalid grievance states, ensuring secure and reliable role-based operations (checklist: Exceptions, User-Defined Exceptions).
- **Sequence Integration:** `assignVendor()` and `selectQuotation()` are called during issue processing, while `resolveGrievance()` and `generateReport()` support grievance redressal and analytics, ensuring seamless integration with Phase 2 workflows (checklist: Consistency with Method in Java Code).

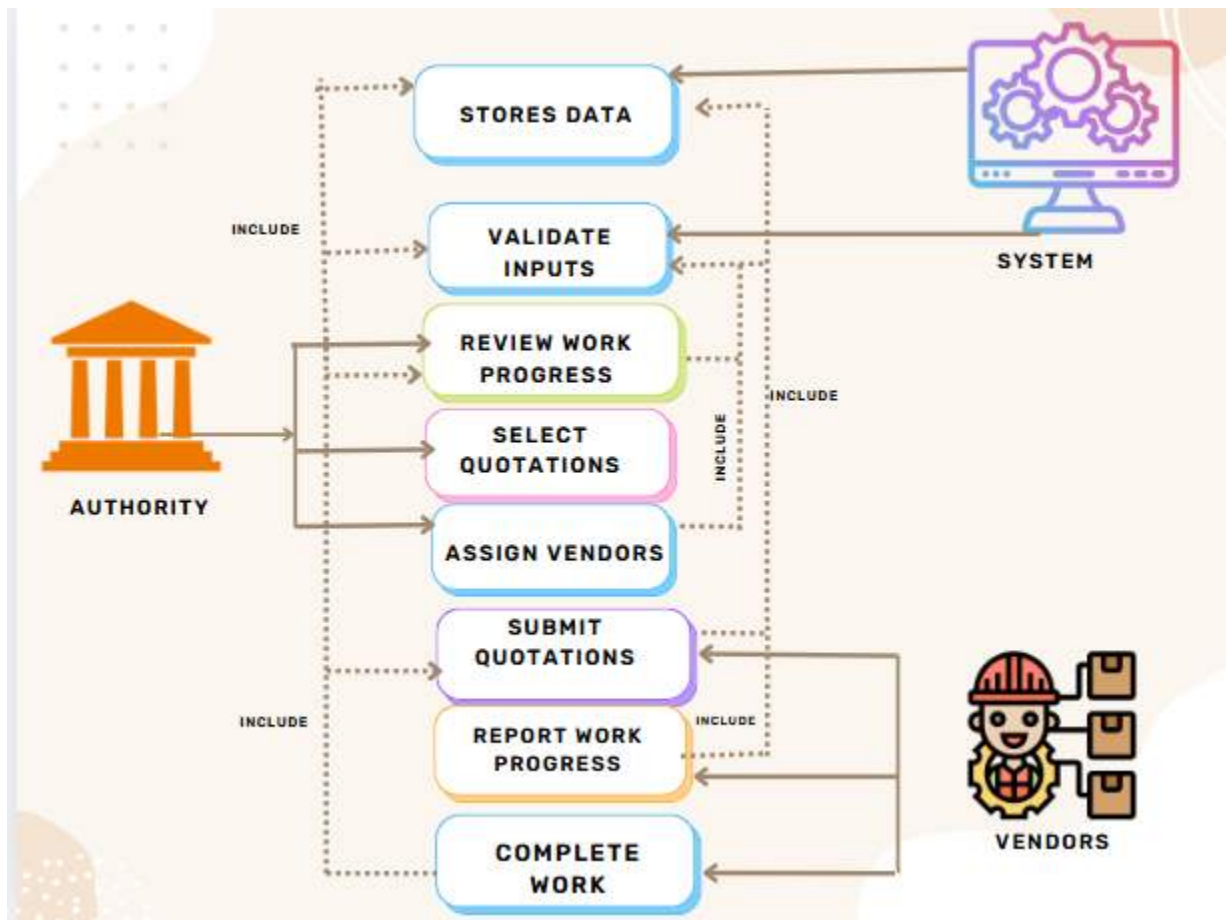
---

## Implementation Details (Alignment with OOP Features)

- **Objects & Classes:**
  - Classes: Citizen, Authority, Administrator, Issue, Vendor, Quotation, Assignment, Grievance, Notification, CivicPulseSystem.
  - Objects represent real-world entities (e.g., a Vendor object per registered vendor).
- **Access Specifiers (Encapsulation):**
  - Private attributes (e.g., price, contactInfo, issued) with public getters/setters for controlled access.
- **Constructors:**
  - Parameterized constructors (e.g., Issue(String description, String location)) and default constructors for flexibility.
- **Exception Handling:**
  - Custom exceptions (e.g., QuotationDeadlineExceededException, VendorCapacityFullException) with try-catch blocks for graceful error management.
- **Collections:**
  - ArrayList for dynamic lists (e.g., issues, quotations, assignments).
  - HashMap for fast lookups (e.g., vendors).
  - LinkedList for ordered notifications.
  - PriorityQueue for prioritized notification processing.
- **Inheritance:**
  - Multilevel inheritance: User → Authority → Administrator.
  - Hierarchical inheritance: Vendor → InfrastructureVendor/SanitationVendor.
- **Abstraction:**
  - Abstract class: User with performAction().
  - Interfaces: Manageable, Reportable for role-based contracts.
- **Compile-Time Polymorphism:**
  - Overloaded methods (e.g., submitIssue(String desc) vs. submitIssue(String desc, String attachment)).
  - Overloaded methods in Quotation (e.g., reviseQuotation(double, int) vs. reviseQuotation(double, int, String)).



## USE CASE DIAGRAM



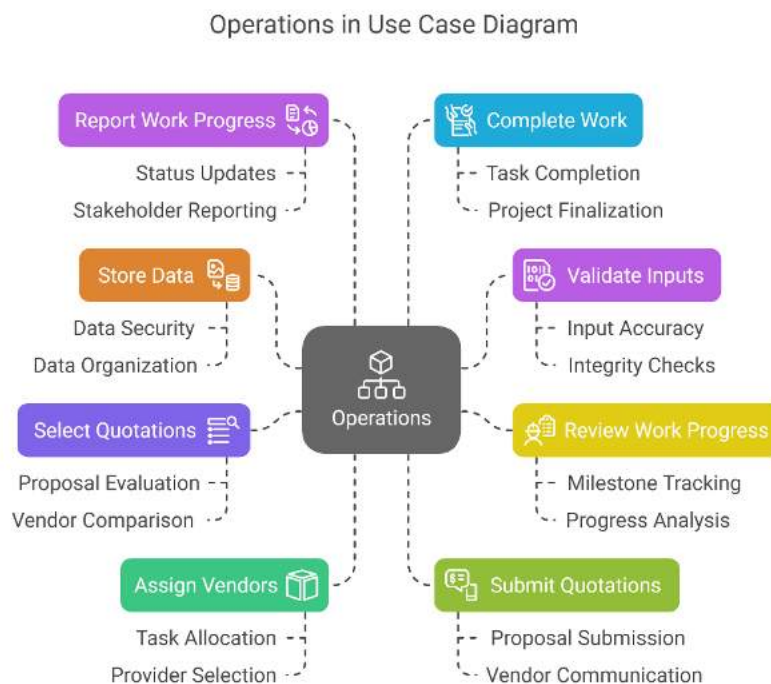
## Overview

### Actors:

- **Authority:** Represents administrative or governmental entities overseeing the resolution process.
- **Vendor:** Represents service providers who execute work on assigned issues.
- **System:** The "Civic Pulse" system that facilitates interactions, validations, and data persistence.

## Operations (from the Use Case Diagram):

1. Store Data
2. Validate Inputs
3. Review Work Progress
4. Select Quotations
5. Assign Vendors
6. Submit Quotations
7. Report Work Progress
8. Complete Work



## Relationships:



- **Includes:** Indicate mandatory sub-tasks (e.g., "Validate Inputs" and "Store Data" are included in multiple use cases).
  - No **extends** relationships are present in the diagram, so the focus is on **include** relationships.
- 

## Use Case Descriptions for Phase 2

### 1. Use Case: Store Data

#### Actors:

- System: The "Civic Pulse" system responsible for data persistence.

#### Preconditions:

- The "Civic Pulse" system must be online and operational.
- Data to be stored (e.g., issue status, quotations, progress updates) must be provided by another operation.
- The target file (e.g., Issues.txt, Quotations.txt) must be accessible for writing.

#### Main Flow:

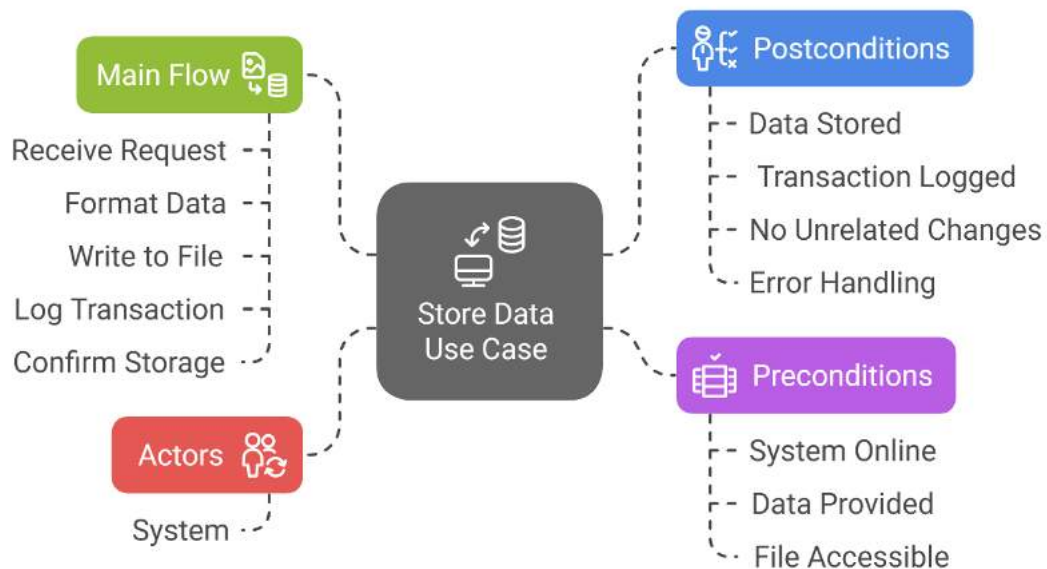
1. The System receives a request to store data from another operation (e.g., "Assign Vendors", "Submit Quotations").
2. The System formats the data into a predefined structure (e.g., "ISSUE001 | Status: Quotations Requested").
3. The System writes the data to the appropriate file (e.g., Issues.txt, Quotations.txt).
4. The System logs the transaction for auditing purposes.
5. The System confirms the data has been stored successfully to the calling operation.

#### Postconditions:

- The data is successfully stored in the designated file.

- The transaction is logged for record-keeping.
- No changes are made to unrelated data.
- If the file is inaccessible, an error message is logged (e.g., "Error saving to Issues.txt"), and the calling operation is notified of the failure.

### Use Case: Store Data in Civic Pulse System



## 2. Use Case: Validate Inputs

**Actors:**

- System: The "Civic Pulse" system responsible for input validation.

**Preconditions:**

- The "Civic Pulse" system must be online and operational.
- Input data (e.g., price, progress percentage) must be provided by another operation (e.g., "Submit Quotations", "Report Work Progress").

**Main Flow:**

1. The System receives input data from another operation.
2. The System checks the data against predefined rules (e.g., price must be positive, progress percentage must be between 0 and 100).
3. If the data is valid, the System allows the calling operation to proceed.
4. If the data is invalid, the System throws an `IllegalArgumentException` with a specific message (e.g., "Price must be positive").

**Postconditions:**

- Valid inputs are passed to the calling operation for further processing.
  - Invalid inputs result in an exception, halting the calling operation.
  - The transaction is logged for record-keeping.
  - No changes are made to the system data.
- 

**3. Use Case: Assign Vendors****Actors:**

- Authority: The administrative entity assigning vendors to issues.

**Preconditions:**

- The Authority must be authenticated with administrative credentials.
- The "Civic Pulse" system must be online and operational.

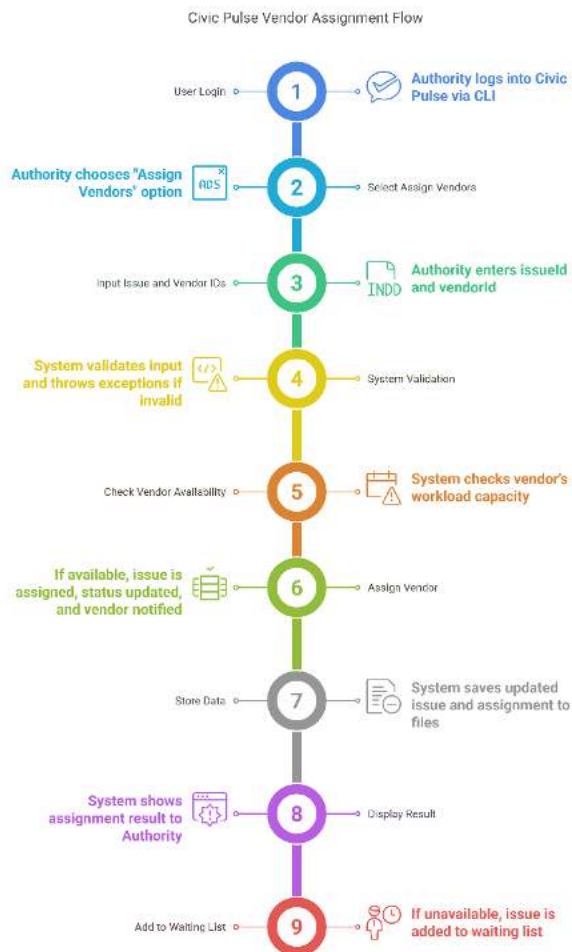
- The Authority must provide a valid issueId for an existing issue and a valid vendorId for an existing vendor.
- The issue must be in an "Open" state (not yet assigned).

**Main Flow:**

1. The Authority logs into the "Civic Pulse" system via CLI.
2. The Authority selects the "Assign Vendors" option.
3. The Authority enters the issueId and vendorId.
4. The System validates the input (e.g., existing issueId, existing vendorId, issue in "Open" state) and throws IllegalArgumentException if invalid.
5. If valid, the System checks the vendor's availability (e.g., workload capacity).
  - If the vendor is available:
    - The System updates the issue status to "Quotations Requested".
    - The System creates an assignment linking the issue to the vendor.
    - The System includes the "Store Data" operation to save the updated issue and assignment to files (e.g., Issues.txt, Assignments.txt).
    - The System notifies the vendor (e.g., "Assigned to ISSUE001").
    - The System displays the result to the Authority (e.g., "Vendor VEND001 assigned to Issue ISSUE001").
  - If the vendor is unavailable:
    - The System adds the issue to a waiting list.
    - The System throws a VendorUnavailableException (e.g., "Vendor capacity exceeded").

**Postconditions:**

- The issue is assigned to the specified vendor, and its status is updated to "Quotations Requested".
- An assignment record is created linking the issue and vendor.
- The transaction is logged, and data is stored in files.
- If the vendor is unavailable, the issue is placed on a waiting list.
- If validation fails, an error message is displayed (e.g., "Invalid Issue ID"), and no assignment occurs.



#### 4. Use Case: Submit Quotations

**Actors:**

- Vendor: The service provider submitting a quotation for an assigned issue.

**Preconditions:**

- The Vendor must be authenticated with valid credentials.
- The "Civic Pulse" system must be online and operational.
- The Vendor must provide a valid issued for an issue assigned to them.
- The issue must be in the "Quotations Requested" state.
- The Vendor must provide a valid price (positive value) and estimated days (positive integer).

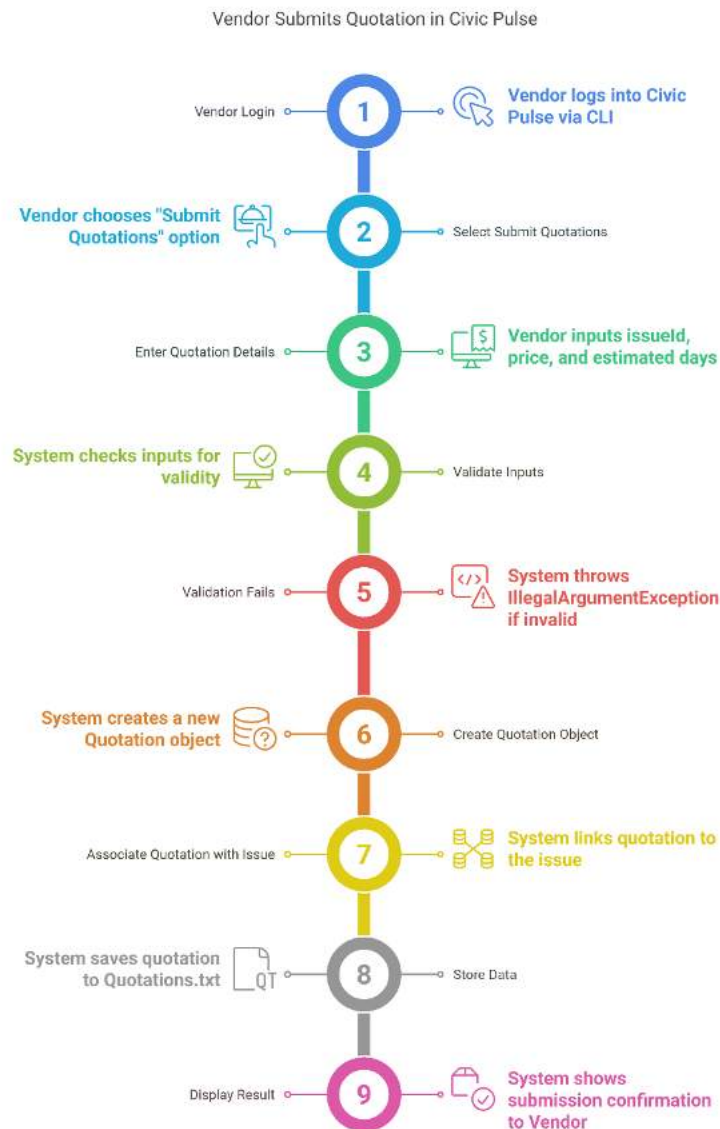
**Main Flow:**

1. The Vendor logs into the "Civic Pulse" system via CLI.
2. The Vendor selects the "Submit Quotations" option.
3. The Vendor enters the issued, price, and estimated days.
4. The System includes the "Validate Inputs" operation to check the inputs (e.g., existing issued, issue in "Quotations Requested" state, price > 0, estimated days > 0).
5. If validation fails, the System throws an IllegalArgumentException (e.g., "Price must be positive").
6. If valid:
  - The System creates a new Quotation object with a unique quotationId.
  - The System associates the quotation with the issue.
  - The System includes the "Store Data" operation to save the quotation to a file (e.g., Quotations.txt).
  - The System displays the result to the Vendor (e.g., "Quotation Q001 submitted for Issue ISSUE001").

**Postconditions:**

- A new quotation is created and associated with the issue.
- The transaction is logged, and the quotation data is stored in a file.
- No changes are made to other data.

- If validation fails, an error message is displayed (e.g., "Issue not found or not ready for quotation"), and no quotation is created.



## 5. Use Case: Select Quotations

**Actors:**

- Authority: The administrative entity selecting a quotation for an issue.

**Preconditions:**

- The Authority must be authenticated with administrative credentials.
- The "Civic Pulse" system must be online and operational.
- The Authority must provide a valid issueId for an existing issue.
- The issue must have at least one submitted quotation and be in the "Quotations Requested" state.

**Main Flow:**

1. The Authority logs into the "Civic Pulse" system via CLI.
2. The Authority selects the "Select Quotations" option.
3. The Authority enters the issueId.
4. The System validates the input (e.g., existing issueId, issue in "Quotations Requested" state, quotations available) and throws IllegalArgumentException if invalid.
5. If valid:
  - The System retrieves all quotations for the issue.
  - The System identifies the quotation with the lowest price (or applies other criteria if specified).
  - The System displays the quotations to the Authority (e.g., "Quotation Q001: \$100, 5 days").
  - The Authority confirms the selection by providing the quotationId.
  - The System updates the issue status to "Vendor Selected".
  - The System notifies the selected vendor (e.g., "Quotation selected for ISSUE001").
  - The System includes the "Store Data" operation to save the updated issue to a file (e.g., Issues.txt).
  - The System displays the result to the Authority (e.g., "Quotation Q001 selected for Issue ISSUE001").

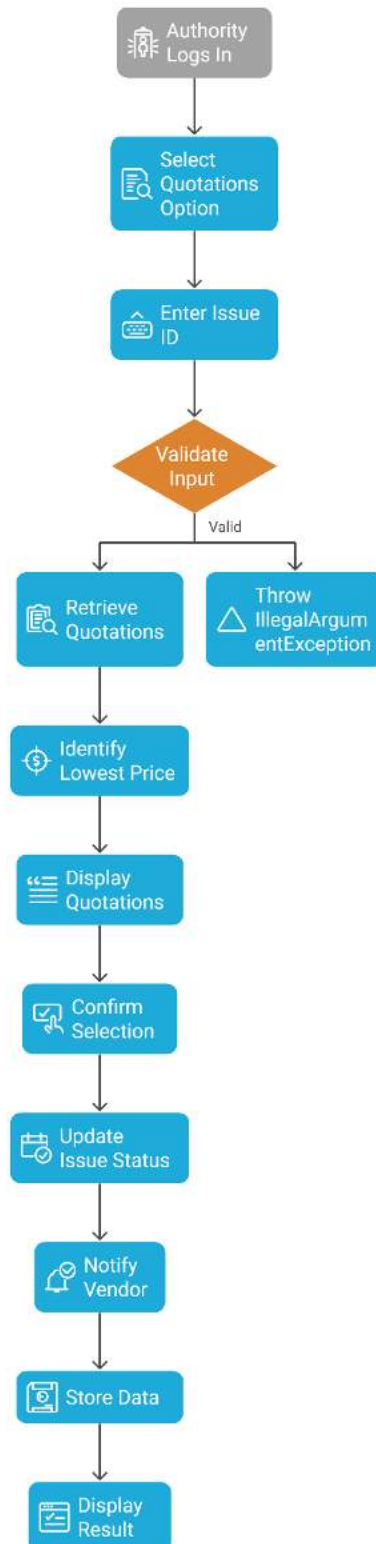
**Postconditions:**

- The issue status is updated to "Vendor Selected", and the selected quotation is marked.



- The transaction is logged, and the updated issue data is stored in a file.
  - The selected vendor is notified.
  - If validation fails, an error message is displayed (e.g., "No quotations available"), and no selection occurs.
-

### Quotation Selection Process



## **6. Use Case: Report Work Progress**

### **Actors:**

- Vendor: The service provider reporting progress on an assigned issue.

### **Preconditions:**

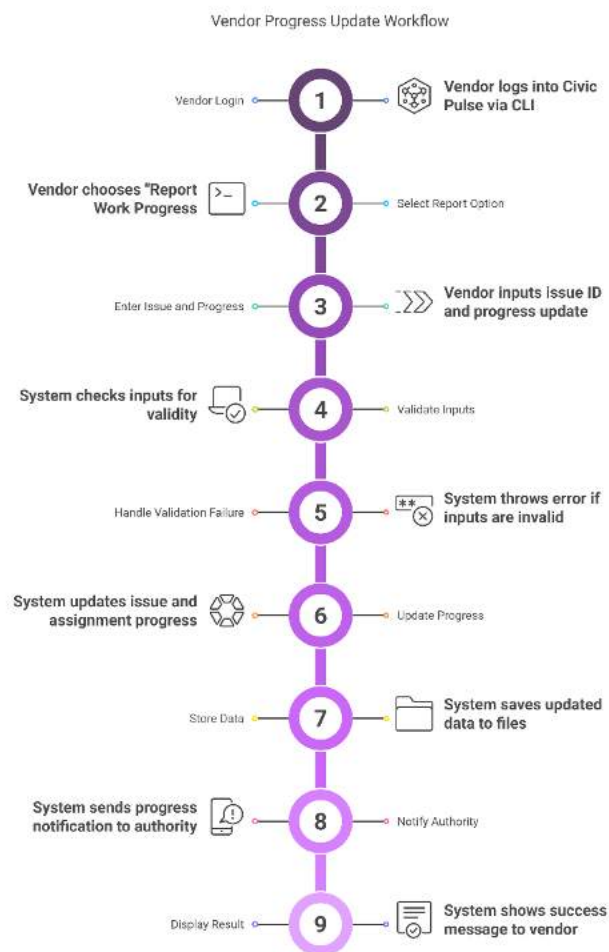
- The Vendor must be authenticated with valid credentials.
- The "Civic Pulse" system must be online and operational.
- The Vendor must provide a valid issuelid for an issue assigned to them.
- The issue must be in the "Vendor Selected" or "Work in Progress" state.
- The Vendor must provide a valid progress update (e.g., percentage between 0 and 100).

### **Main Flow:**

1. The Vendor logs into the "Civic Pulse" system via CLI.
2. The Vendor selects the "Report Work Progress" option.
3. The Vendor enters the issuelid and progress update (e.g., "50% complete").
4. The System includes the "Validate Inputs" operation to check the inputs (e.g., existing issuelid, issue in correct state, valid progress percentage).
5. If validation fails, the System throws an IllegalArgumentException (e.g., "Progress must be between 0 and 100").
6. If valid:
  - The System updates the progress in the associated assignment and issue.
  - The System includes the "Store Data" operation to save the updated issue and assignment to files (e.g., Issues.txt, Assignments.txt).
  - The System notifies the Authority (e.g., "ISSUE001: 50% complete").
  - The System displays the result to the Vendor (e.g., "Progress updated for Issue ISSUE001").

## Postconditions:

- The issue and assignment records are updated with the new progress.
- The transaction is logged, and the updated data is stored in files.
- The Authority is notified of the progress update.
- If validation fails, an error message is displayed (e.g., "Issue not assigned to vendor"), and no update occurs.



## 7. Use Case: Review Work Progress

### Actors:

- Authority: The administrative entity reviewing the progress of an issue.

### Preconditions:

- The Authority must be authenticated with administrative credentials.
- The "Civic Pulse" system must be online and operational.
- The Authority must provide a valid issued for an existing issue.
- The issue must be in the "Work in Progress" state.

### Main Flow:

1. The Authority logs into the "Civic Pulse" system via CLI.
2. The Authority selects the "Review Work Progress" option.
3. The Authority enters the issued.
4. The System validates the input (e.g., existing issued, issue in "Work in Progress" state) and throws `IllegalArgumentException` if invalid.
5. If valid:
  - The System retrieves the current progress from the issue.
  - The System displays the progress to the Authority (e.g., "Issue ISSUE001: 50% complete").
  - The System includes the "Store Data" operation to log the review in a file (e.g., `ProgressLog.txt`).

### Postconditions:

- The Authority is informed of the current progress of the issue.
- The transaction is logged, and the review is stored in a file.
- No changes are made to the issue data.
- If validation fails, an error message is displayed (e.g., "Issue not in progress"), and no review occurs.

### Tracking Work Progress in Civic Pulse

**The Authority logs into the "Civic Pulse" system via CLI**

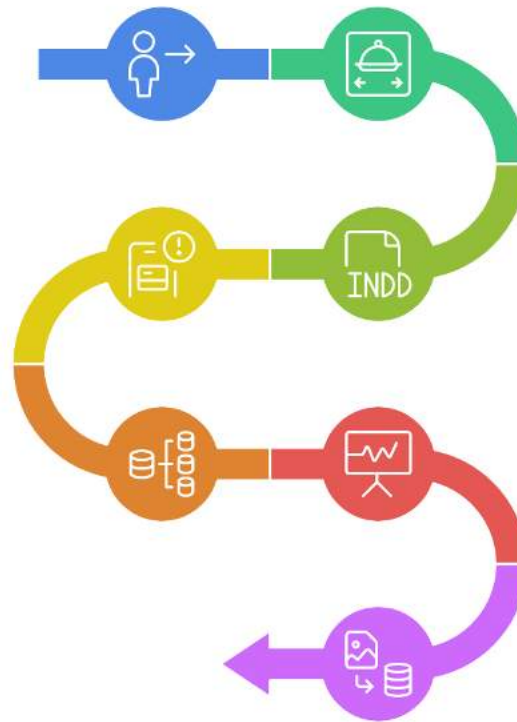
User login interface

**The System validates the input**

Validation check process

**The System retrieves the current progress from the issue**

Data retrieval process



**The Authority selects the "Review Work Progress" option**

Menu selection screen

**The Authority enters the issued**

Input field for issue ID

**The System displays the progress to the Authority**

Progress display screen

**The System includes the "Store Data" operation to log the review in a file**

Data logging process

## 8. Use Case: Complete Work

### Actors:

- Vendor: The service provider marking an issue as completed.

### Preconditions:

- The Vendor must be authenticated with valid credentials.
- The "Civic Pulse" system must be online and operational.
- The Vendor must provide a valid issued for an issue assigned to them.
- The issue must be in the "Work in Progress" state.
- The Vendor must provide completion evidence (e.g., a text description of the work done).

### Main Flow:

1. The Vendor logs into the "Civic Pulse" system via CLI.
2. The Vendor selects the "Complete Work" option.
3. The Vendor enters the issued and completion evidence.
4. The System validates the input (e.g., existing issued, issue in "Work in Progress" state) and throws `IllegalArgumentException` if invalid.
5. If valid:
  - The System updates the assignment status to "Completed".
  - The System updates the issue status to "Completed".
  - The System records the completion evidence in the issue.
  - The System includes the "Store Data" operation to save the updated issue and assignment to files (e.g., `Issues.txt`, `Assignments.txt`).
  - The System notifies the Authority (e.g., "ISSUE001 completed, pending review").
  - The System displays the result to the Vendor (e.g., "Work completed for Issue ISSUE001").

### Postconditions:

- The issue and assignment are marked as "Completed".
- The completion evidence is recorded.

- The transaction is logged, and the updated data is stored in files.
- The Authority is notified for final review.
- If validation fails, an error message is displayed (e.g., "Work not in progress"), and no completion occurs.





## **Class Diagram**

[https://drive.google.com/drive/folders/1IYKXuB1Sn1fcL98POS7hUu0EuheJGvi7?usp=drive link](https://drive.google.com/drive/folders/1IYKXuB1Sn1fcL98POS7hUu0EuheJGvi7?usp=drive_link)

## **Overview of Combined Class Diagram (Phase 1 + Phase 2)**

### **Purpose:**

The Class Diagram represents the structure of the "Civic Pulse" system, a platform designed to facilitate civic engagement and issue resolution across two phases. Phase 1 focuses on user registration, issue reporting, proposal submission, voting, commenting, dashboard viewing, grievance filing, and issue review/prioritization by Citizens and Authorities. Phase 2 focuses on the solution process, including vendor assignment, quotation submission, work execution, progress reporting, completion, and grievance resolution during work, managed by Authorities and Vendors. The diagram includes classes for users (e.g., Citizen, Authority, Vendor), engagement entities (e.g., Issue, Proposal, Quotation, Assignment), system management (CivicPulseSystem), and supporting entities (e.g., Address, Comment, Notification, Grievance), along with interfaces (Engageable, Trackable). The design leverages abstraction, inheritance, composition, and interfaces to ensure reusability, extensibility, and maintainability.

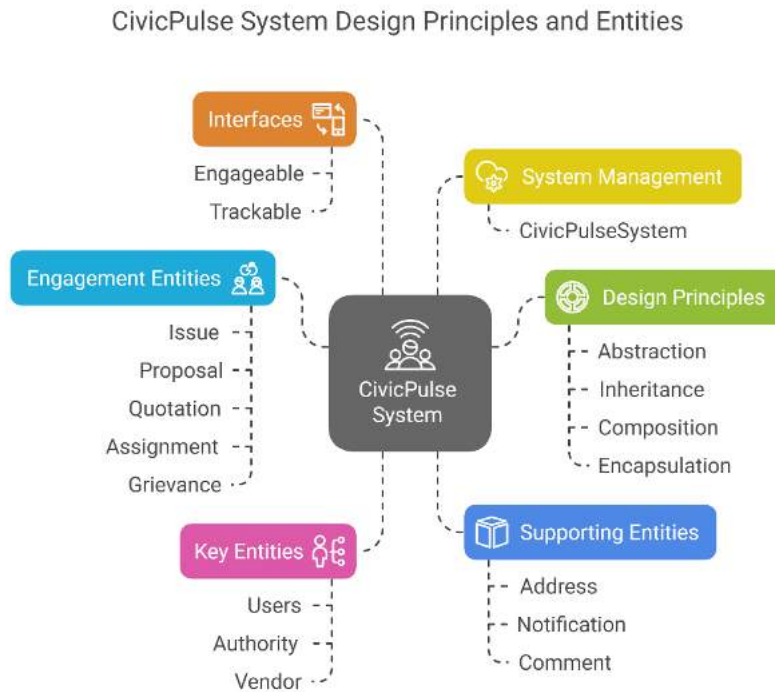
### **Key Entities:**

- **Users:** Citizen (Phase 1), Authority (Phase 1 & 2), Vendor (Phase 2).
- **Engagement Entities:** Issue (Phase 1 & 2), Proposal (Phase 1), Quotation (Phase 2), Assignment (Phase 2), Grievance (Phase 1 & 2).
- **System Management:** CivicPulseSystem (Phase 1 & 2).
- **Supporting Entities:** Address (Phase 1), Notification (Phase 1 & 2), Comment (Phase 1).
- **Interfaces:** Engageable (voting, commenting), Trackable (tracking status).

### **Design Principles:**

- **Abstraction:** Interfaces like Engageable and Trackable define common behaviors.

- **Inheritance:** User hierarchy (User → Citizen/Authority/Vendor).
- **Composition:** Classes like Issue and Proposal contain lists of comments, votes, etc.
- **Encapsulation:** Private attributes with public getters/setters for controlled access.



### Types of Arrows and Their Meanings :-

The Class Diagram for the "Civic Pulse" system uses several types of arrows to represent relationships between classes, interfaces, and entities. Below are the types of arrows used in the diagram and their meanings:

#### 1. **Inheritance (Solid Line with Hollow Arrowhead: <|--)**

- **Example:** User <|-- Citizen, User <|-- Authority, User <|-- Vendor
- **Meaning:** Represents an "is-a" relationship where a subclass inherits attributes and methods from a superclass. For instance, Citizen, Authority, and Vendor

inherit from the abstract User class, meaning they are specific types of users sharing common attributes (e.g., `userId`, `name`) and methods (e.g., `authenticate()`). This ensures code reuse and establishes a hierarchical structure.

## 2. Interface Implementation (Dashed Line with Hollow Arrowhead: `.->`)

- **Example:** `Issue .-> Engageable`, `Issue .-> Trackable`, `Proposal .-> Engageable`, `Proposal .-> Trackable`
- **Meaning:** Indicates that a class implements an interface, adhering to its contract. For example, `Issue` and `Proposal` implement the `Engageable` interface (which defines methods like `vote()` and `comment()`) and the `Trackable` interface (which defines methods like `setStatus()`). This ensures consistent behavior across classes while allowing flexibility in implementation.

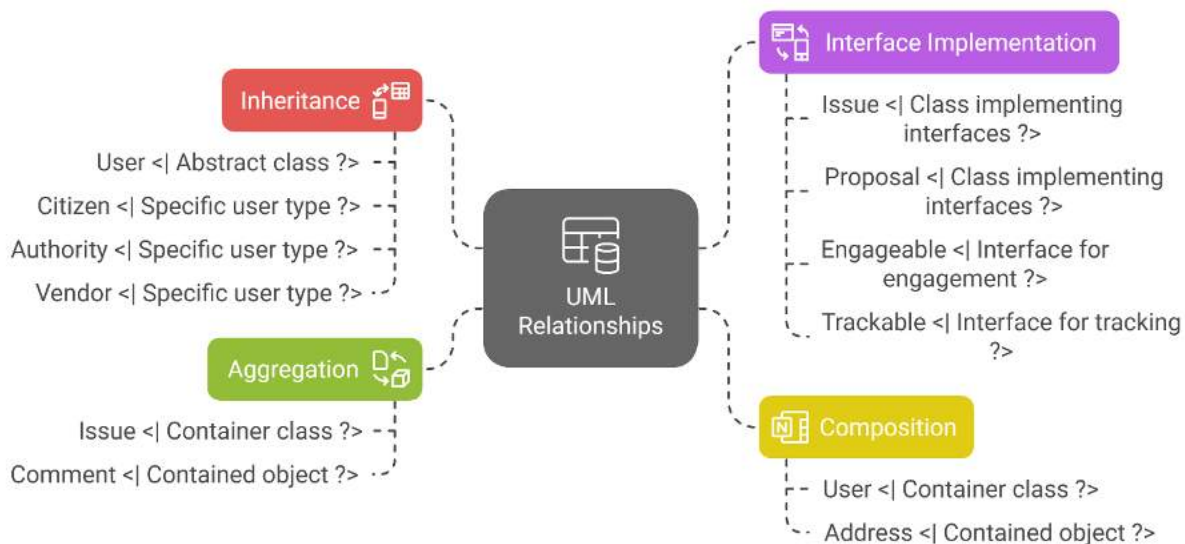
## 3. Composition (Solid Line with Filled Diamond: `o-->`)

- **Example:** `User "1" --> "1" Address : has`
- **Meaning:** Represents a "has-a" relationship where one class contains an instance of another class as a part, and the contained instance cannot exist independently of the container. In this case, each `User` has exactly one `Address`, and the `Address` is tightly coupled to the `User` (if the `User` is destroyed, so is the `Address`). This models the real-world relationship where a user's address is an integral part of their profile.

## 4. Aggregation (Solid Line with Hollow Diamond: `<-->`)

- Not explicitly used in this diagram but worth noting for completeness. Aggregation represents a "has-a" relationship where the contained objects can exist independently of the container. For example, if `Issue` had a list of `Comment` objects as an aggregation (instead of composition), the comments could exist independently of the issue.

## UML Relationships in Software Design



### 5. Association (Solid Line: -->)

- **Example:** Citizen "1" --> "0..\*" Issue : reports, Authority "1" --> "0..\*" Issue : reviews, Vendor "1" --> "0..\*" Quotation : submits
- **Meaning:** Represents a general relationship between two classes, indicating that one class interacts with or uses another. For instance, a Citizen can report multiple Issue objects (1-to-many relationship), an Authority can review multiple Issue objects, and a Vendor can submit multiple Quotation objects. The multiplicities (e.g., 0..\*) specify the cardinality of the relationship.

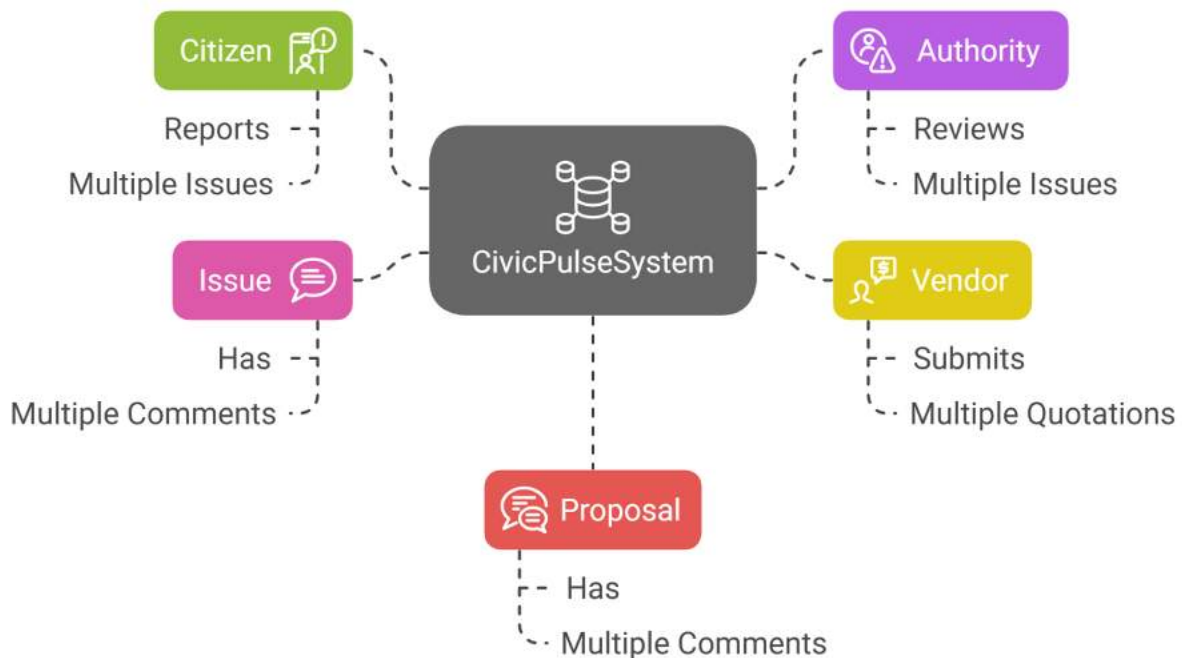
### 6. Directed Association (Solid Line with Arrow: -->)

- **Example:** CivicPulseSystem --> Citizen, CivicPulseSystem --> Issue
- **Meaning:** Indicates a directional relationship where one class (e.g., CivicPulseSystem) references or manages instances of another class (e.g., Citizen, Issue). This is used to show that CivicPulseSystem is the central coordinator that manages collections of various entities like citizens, issues, and quotations.

### 7. Composition with Multiplicity (Solid Line with Filled Diamond and Multiplicity)

- **Example:** Issue "1" --> "0..\*" Comment : has, Proposal "1" --> "0..\*" Comment : has
- **Meaning:** Similar to composition, but with specified multiplicity. An Issue or Proposal can have zero or more Comment objects, and these comments are part of the issue/proposal (if the issue/proposal is destroyed, the comments are too). This models the relationship where comments are intrinsically tied to their parent entity.

### Relationships in CivicPulseSystem



## Use of Abstract Classes

Abstract classes define common blueprints for related classes, enforcing a structure that subclasses must adhere to while allowing shared behavior. In this diagram, one class is abstract:

- **User (Abstract):**
  - **Attributes:** -userId: String, -name: String, -email: String, -password: String, -address: Address.
  - **Methods:** +authenticate(name: String, password: String): boolean, +getUserId(): String, +getName(): String, +setEmail(email: String).
  - **Justification:** The User class is abstract as it represents a generic user entity with core attributes (e.g., identification, credentials, address) and behaviors (e.g., authentication). The specific implementation varies between Citizen, Authority, and Vendor (e.g., different roles and permissions). Abstraction ensures subclasses provide concrete implementations, promoting polymorphism and reducing redundancy. It also supports future user types (e.g., Moderator) without altering the core design.

## Use of Inheritance

Inheritance establishes a hierarchical relationship where subclasses inherit attributes and methods from superclasses, promoting code reuse and an "is-a" relationship. The diagram employs single-level inheritance:

- **Single-Level Inheritance:**
  - **User <|-- Citizen, User <|-- Authority, User <|-- Vendor.**
  - **Justification:** Citizen, Authority, and Vendor are specific types of User, inheriting common attributes (e.g., userId, password) and methods (e.g., authenticate()). This reduces redundancy, with user-related data and authentication logic defined

once in User. Citizen adds methods like `reportIssue()`, `submitProposal()`, and `voteOnIssueOrProposal()` for civic participation (Phase 1), Authority adds `reviewIssue()`, `prioritizeIssue()` (Phase 1), and `assignVendors()`, `selectQuotations()` (Phase 2) for administrative roles, and Vendor adds `submitQuotation()`, `reportWorkProgress()`, and `completeWork()` for solution execution (Phase 2). Inheritance ensures a clear role-based hierarchy built on the generic User foundation.

### **Additional Design Elements**

- **Interfaces (Engageable, Trackable):**
  - Issue and Proposal realize Engageable (with methods like `vote()`, `comment()`) and Trackable (with methods like `setStatus()`, `getStatus()`) to enforce these behaviors, promoting a contract-based design for consistency and extensibility across engagement entities in Phase 1. This ensures that all engagement entities support voting, commenting, and status tracking in a standardized way.
- **Composition (User o--> Address, Issue o--> Comment, Proposal o--> Comment):**
  - User has an Address object, encapsulating details (e.g., country, state), which supports data organization and reusability for user registration in Phase 1. Issue and Proposal have lists of Comment objects, modeling the relationship where comments are part of issues/proposals and cannot exist independently, supporting community engagement features in Phase 1.
- **Associations (CivicPulseSystem to Citizen, Issue, Vendor, etc.):**
  - CivicPulseSystem manages collections of these entities, reflecting its role as the system coordinator across both phases. For example, it manages Citizen and Issue for Phase 1 operations (e.g., reporting issues) and Vendor and Quotation for Phase 2 operations (e.g., submitting quotations).
- **Phase 2-Specific Relationships:**
  - Issue has lists of Quotation and one Assignment, and Vendor submits Quotation and executes Assignment, modeling the solution workflow where issues are assigned to vendors, who submit quotations and perform work.
- **Exception Handling:**
  - Methods in CivicPulseSystem (e.g., `reportIssue()`, `submitQuotation()`) include implicit exception handling (e.g., `IllegalArgumentException` for invalid inputs),

preparing for robust error handling in implementation, as specified in the use cases for both phases.

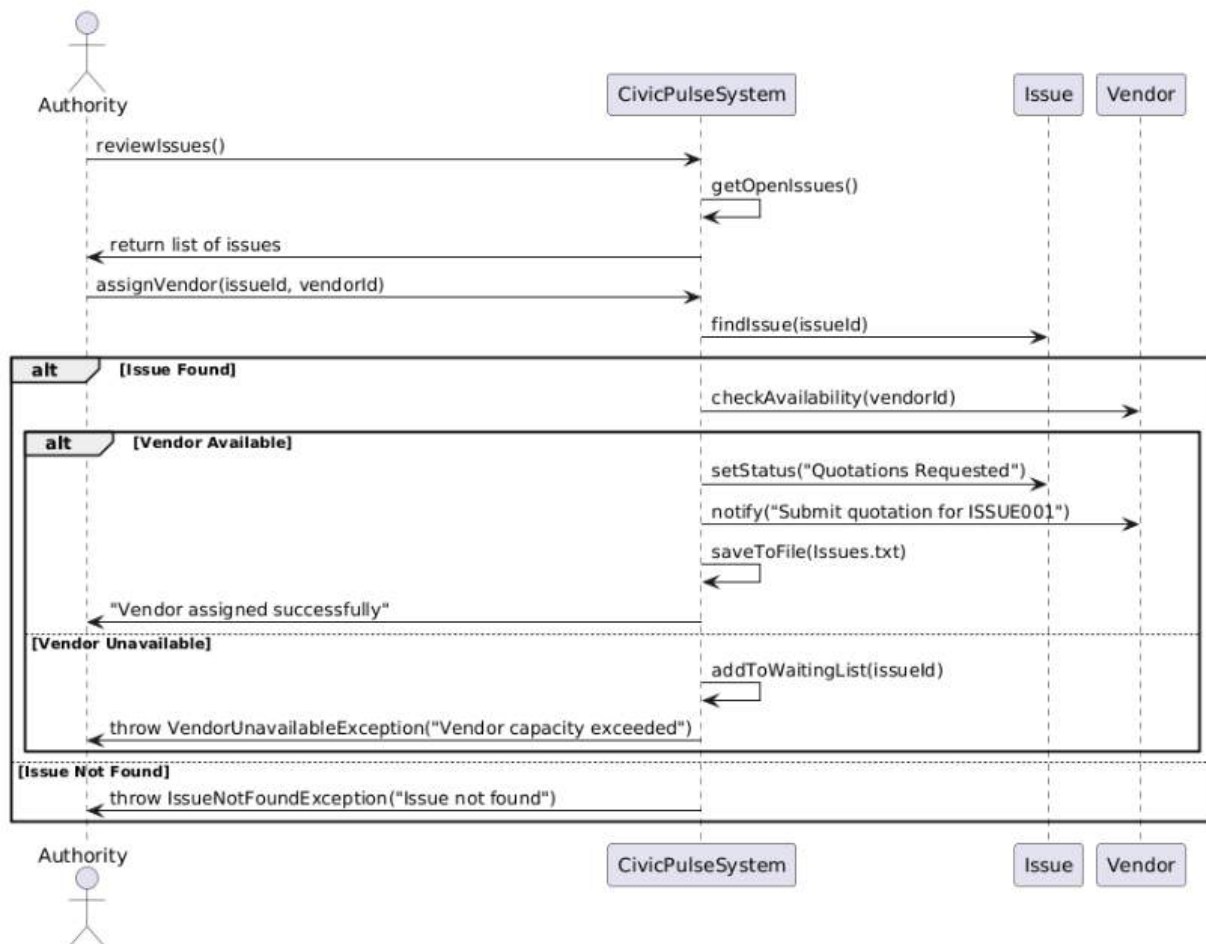
### Why This Design Works

- **Abstraction:** By making User abstract, the design enforces a contract that subclasses (Citizen, Authority, Vendor) must fulfill, reducing errors and supporting future expansion (e.g., new user roles).
  - **Inheritance:** The single-level inheritance hierarchy (User to Citizen, Authority, Vendor) minimizes code duplication and aligns with the system's domain (e.g., citizens, authorities, and vendors as users).
  - **Flexibility:** Interfaces (Engageable, Trackable) and inheritance allow the system to evolve (e.g., new engagement types like petitions, new user roles like moderators) without major redesign.
  - **Encapsulation:** Private attributes with public methods (e.g., getUserId(), setStatus()) ensure data integrity and controlled access across all entities.
  - **Maintainability:** The clear hierarchy and separation of concerns (e.g., user management vs. engagement handling vs. solution execution) simplify updates and debugging, supporting both Phase 1 and Phase 2 functionalities.
-



## Sequence Diagram

Sequence Diagram 1: Review Issues and Assign Vendor



### **Purpose:**

This sequence diagram illustrates the process where an Authority reviews open issues and assigns a vendor to address a specific issue, initiating the quotation process.

### **Flow:**

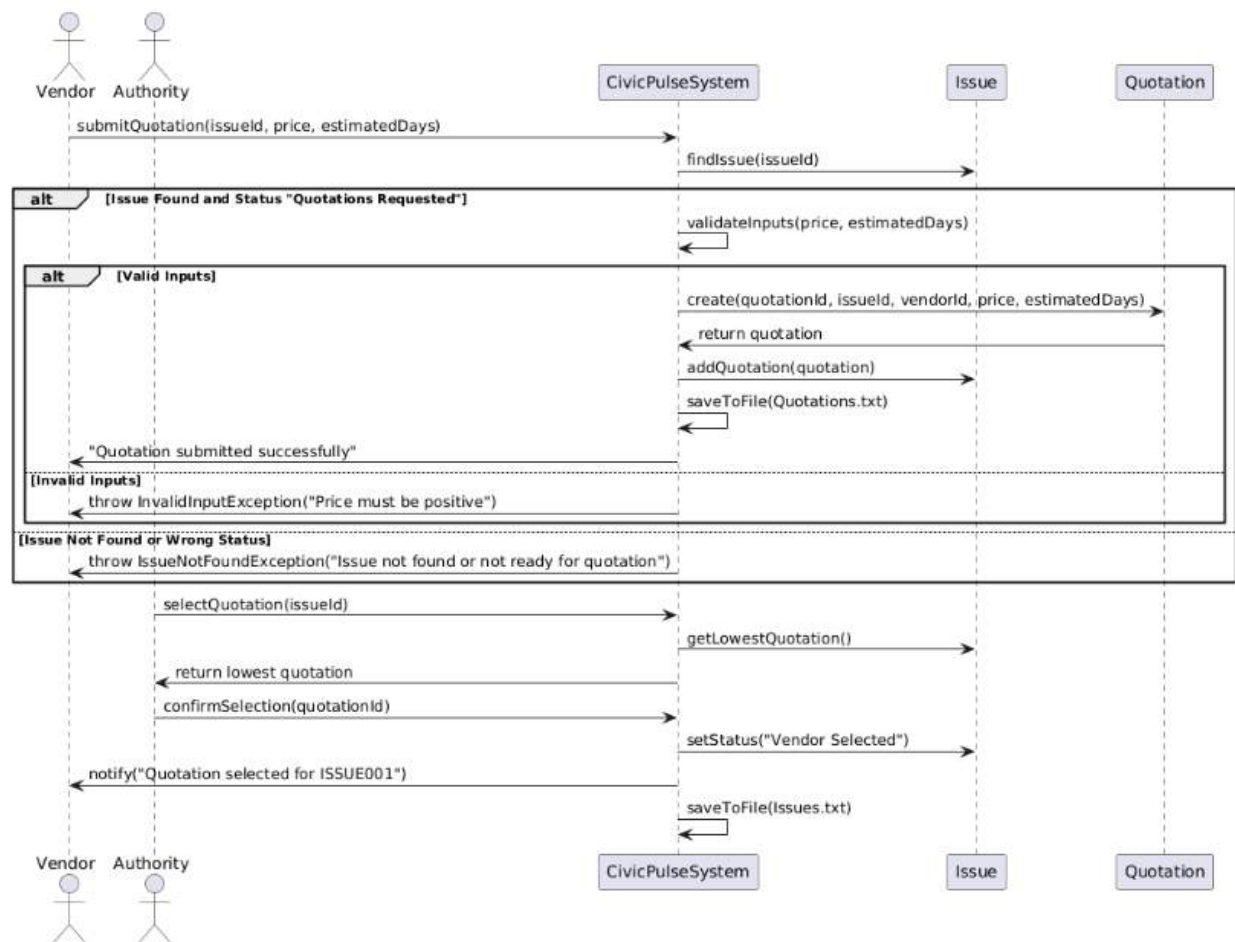
1. The Authority requests to review issues by calling `reviewIssues()` on the CivicPulseSystem.

2. The system retrieves a list of open issues (getOpenIssues()) and returns it to the Authority.
3. The Authority selects an issue and assigns a vendor by calling assignVendor(issueId, vendorId).
4. The system checks if the issue exists (findIssue(issueId)).
  - If the issue is found:
    - The system checks the vendor's availability (checkAvailability(vendorId)).
      - If the vendor is available:
        - The issue status is updated to "Quotations Requested".
        - The vendor is notified to submit a quotation ("Submit quotation for ISSUE001").
        - The updated issue data is saved to a file (saveToFile(Issues.txt)).
        - The Authority receives a success message ("Vendor assigned successfully").
      - If the vendor is unavailable:
        - The issue is added to a waiting list (addToWaitingList(issueId)).
        - An exception is thrown (VendorUnavailableException).
    - If the issue is not found:
      - An exception is thrown (IssueNotFoundException).

**Explanation:**

This sequence ensures that issues are systematically assigned to available vendors, with proper validation to prevent errors (e.g., assigning to a non-existent issue or unavailable vendor). The waiting list functionality handles resource constraints, and saving to a file ensures data persistence, aligning with the "Assign Vendors" use case and its "Store Data" inclusion.

## Sequence Diagram 2: Submit Quotation and Select Quotation



### Purpose:

This sequence diagram shows the process where a Vendor submits a quotation for an assigned issue, and the Authority selects the most suitable quotation to proceed with the work.

### Flow:

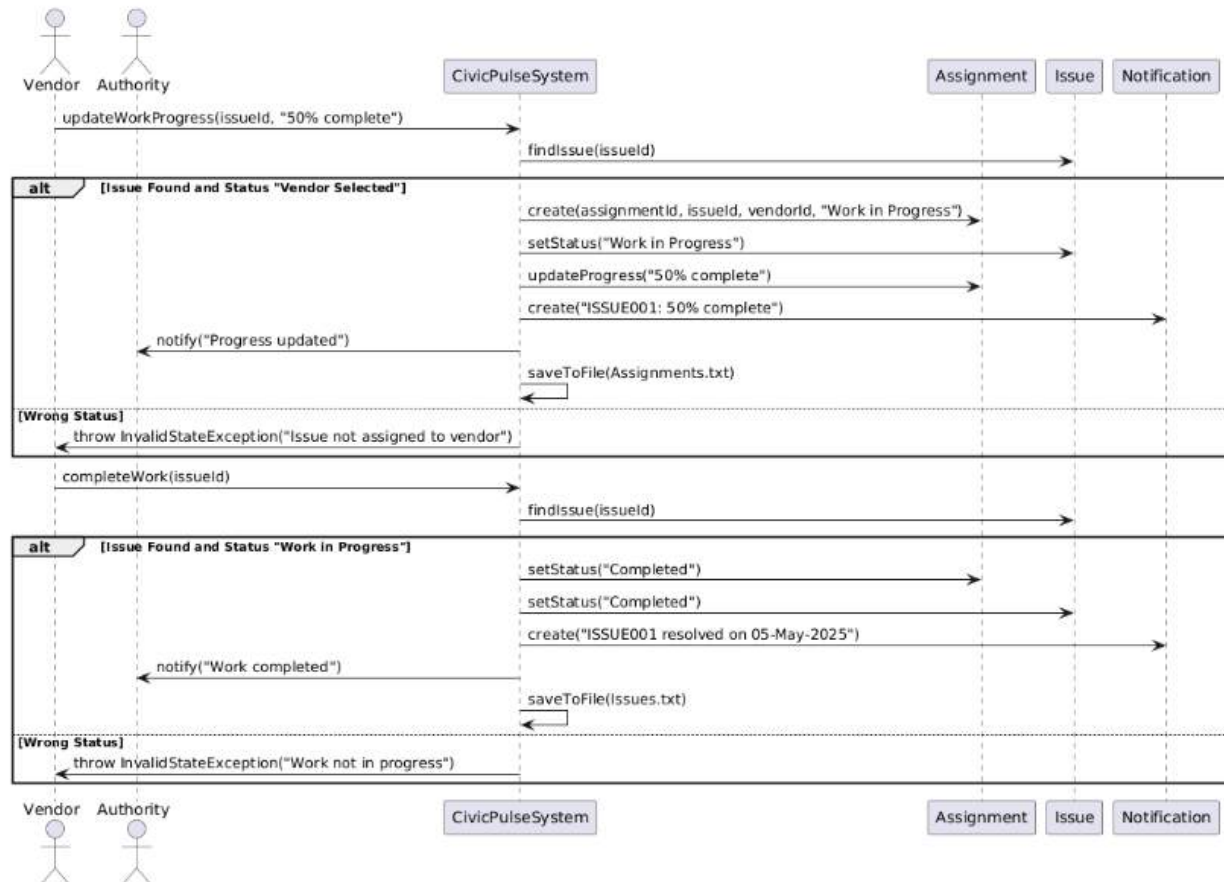
1. The Vendor submits a quotation by calling `submitQuotation(issueId, price, estimatedDays)` on the CivicPulseSystem.

2. The system checks if the issue exists and is in the "Quotations Requested" state (`findIssue(issueId)`).
  - If the issue is found and in the correct state:
    - The system validates the quotation inputs (`validateInputs(price, estimatedDays)`).
      - If inputs are valid:
        - A new Quotation object is created (`create(quotationId, issueId, vendorId, price, estimatedDays)`).
        - The quotation is added to the issue (`addQuotation(quotation)`).
        - The quotation data is saved to a file (`saveToFile(Quotations.txt)`).
        - The Vendor receives a success message ("Quotation submitted successfully").
      - If inputs are invalid:
        - An exception is thrown (`InvalidInputException`).
    - If the issue is not found or not in the correct state:
      - An exception is thrown (`IssueNotFoundException`).
  - 3. The Authority initiates the quotation selection by calling `selectQuotation(issueId)`.
  - 4. The system retrieves the lowest-priced quotation (`getLowestQuotation()`) and returns it to the Authority.
  - 5. The Authority confirms the selection (`confirmSelection(quotationId)`).
  - 6. The system updates the issue status to "Vendor Selected", notifies the Vendor ("Quotation selected for ISSUE001"), and saves the updated issue to a file (`saveToFile(Issues.txt)`).

**Explanation:**

This sequence ensures competitive bidding by allowing vendors to submit quotations, with validation to maintain data integrity (e.g., positive price). The Authority's selection process ensures cost-effectiveness by choosing the lowest quotation, aligning with the "Submit Quotations" and "Select Quotations" use cases. Notifications keep vendors informed, and file storage ensures persistence.

Sequence Diagram 3: Update Work Progress and Complete Work



### Purpose:

This sequence diagram depicts the Vendor updating the progress of an assigned issue and subsequently marking the work as completed, with notifications sent to the Authority.

### Flow:

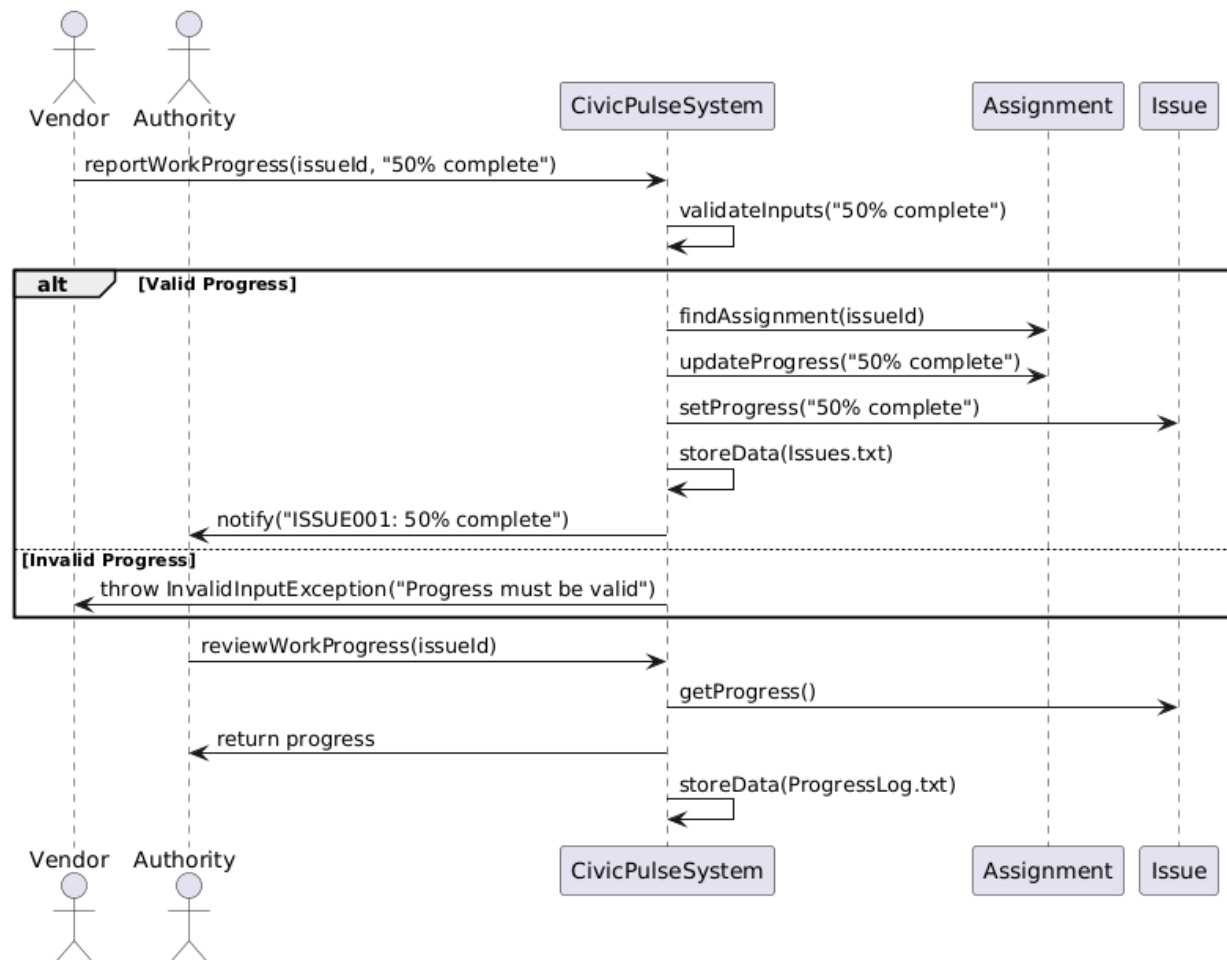
1. The Vendor updates work progress by calling `updateWorkProgress(issueId, "50% complete")` on the `CivicPulseSystem`.
2. The system checks if the issue exists and is in the "Vendor Selected" state (`findIssue(issueId)`).
  - If the issue is found and in the correct state:

- An Assignment is created to track the work (create(assignmentId, issued, vendorId, "Work in Progress")).
  - The issue status is updated to "Work in Progress" (setStatus("Work in Progress")).
  - The progress is updated in the assignment (updateProgress("50% complete")).
  - A notification is created and sent to the Authority ("Progress updated").
  - The assignment data is saved to a file (saveToFile(Assignments.txt)).
  - If the issue is not in the correct state:
    - An exception is thrown (InvalidStateException).
3. The Vendor marks the work as completed by calling completeWork(issued).
4. The system checks if the issue exists and is in the "Work in Progress" state.
- If the issue is found and in the correct state:
    - The assignment status is updated to "Completed" (setStatus("Completed")).
    - The issue status is updated to "Completed" (setStatus("Completed")).
    - A notification is created and sent to the Authority ("Work completed").
    - The updated issue data is saved to a file (saveToFile(Issues.txt)).
  - If the issue is not in the correct state:
    - An exception is thrown (InvalidStateException).

**Explanation:**

This sequence ensures that work progress is tracked and communicated effectively, aligning with the "Report Work Progress" use case (though named differently here). The completion step fulfills the "Complete Work" use case. State validation ensures the workflow progresses logically (e.g., can't update progress unless the vendor is selected), and notifications keep the Authority informed. File storage ensures data persistence.

Sequence Diagram 4: Report Work Progress and Review Work Progress



### Purpose:

This sequence diagram shows the Vendor reporting work progress and the Authority reviewing the progress, ensuring oversight and transparency during issue resolution.

### Flow:

1. The Vendor reports work progress by calling `reportWorkProgress(issueId, "50% complete")` on the `CivicPulseSystem`.
2. The system validates the progress input (`validateInputs("50% complete")`).
  - If the progress is valid:
    - The system locates the assignment (`findAssignment(issueId)`).

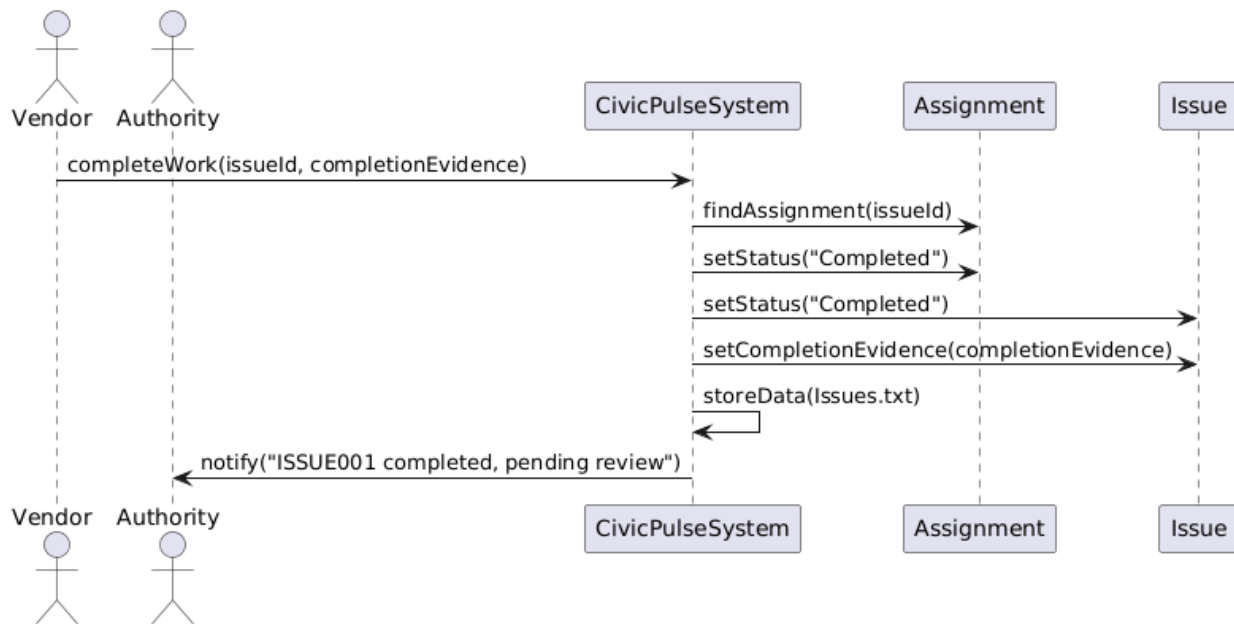
- The progress is updated in the assignment (updateProgress("50% complete")).
- The issue's progress is updated (setProgress("50% complete")).
- The updated data is saved to a file (storeData(Issues.txt)).
- The Authority is notified ("ISSUE001: 50% complete").
- If the progress is invalid:
  - An exception is thrown (InvalidInputException).
- 3. The Authority reviews the progress by calling reviewWorkProgress(issueId).
- 4. The system retrieves the progress from the issue (getProgress()) and returns it to the Authority.
- 5. The system logs the review by saving to a file (storeData(ProgressLog.txt)).

**Explanation:**

This sequence directly aligns with the "Report Work Progress" and "Review Work Progress" use cases, ensuring that progress updates are validated and communicated to the Authority. The Authority's ability to review progress provides oversight, and file storage ensures a record of both updates and reviews, fulfilling the "Store Data" inclusion.

## Sequence Diagram 5: Complete Work





### Purpose:

This sequence diagram illustrates the Vendor marking an issue as completed, with the Authority being notified for final review, concluding the resolution process.

### Flow:

1. The Vendor marks the work as completed by calling `completeWork(issuelid, completionEvidence)` on the `CivicPulseSystem`.
2. The system locates the assignment (`findAssignment(issuelid)`).
3. The assignment status is updated to "Completed" (`setStatus("Completed")`).
4. The issue status is updated to "Completed" (`setStatus("Completed")`).
5. The completion evidence is recorded (`setCompletionEvidence(completionEvidence)`).
6. The updated issue data is saved to a file (`storeData(Issues.txt)`).
7. The Authority is notified ("ISSUE001 completed, pending review").

### Explanation:

This sequence fulfills the "Complete Work" use case, ensuring that the issue resolution is finalized and recorded. The notification to the Authority allows for final oversight (e.g., verification of work quality), and saving to a file aligns with the "Store Data" inclusion. Note that this diagram overlaps with the completion step in Sequence Diagram 3, but it provides a more focused view without the prior progress update.

## **PACKAGES**

### **Package Structure for Civic Pulse System**

The Civic Pulse system includes three roles: **Citizen**, **Authority**, and **Vendor**. The system is divided into two phases:

- **Phase 1:** Focuses on citizen-driven operations, such as issue submission, account management, and grievance filing.
- **Phase 2:** Focuses on vendor and authority operations, such as vendor assignment, quotation management, and work progress tracking.

The package structure is designed to support these roles and phases, ensuring modularity, scalability, and adherence to OOP principles like encapsulation, inheritance, abstraction, and polymorphism. Below is the complete package structure with all details.

---

CivicPulseApp

|— src

```
|  └─ com
|
|    └─ civicpulse
|
|      └─ core
|
|        └─ Category.java
|
|          └─ Role.java
|
|            └─ user
|
|              └─ model
|
|                └─ User.java
|
|                  └─ Citizen.java
|
|                    └─ Authority.java
|
|                      └─ VendorUser.java
|
|                        └─ issue
|
|                          └─ model
|
|                            └─ Issue.java
|
|                              └─ Comment.java
|
|                                └─ proposal
|
|                                  └─ model
|
|                                    └─ Proposal.java
|
|                                      └─ vendor
|
|                                        └─ model
```

```

|      |      |— Vendor.java
|      |      |— Assignment.java
|      |      └— Quotation.java
|      |— grievance
|      |  └— model
|      |      |— Grievance.java
|      └— system
|          |— CivicPulse.java
|          └— CivicPulseGUI.java
|— bin
|  └— com
|      └— civicpulse
|          |— core
|          |  |— Category.class
|          |  └— Role.class
|          |— user
|          |  └— model
|          |      |— User.class
|          |      |— Citizen.class
|          |      |— Authority.class

```

```

|      |      └─ VendorUser.class
|
|      └─ issue
|
|      |      └─ model
|
|      |      └─ Issue.class
|
|      |      └─ Comment.class
|
|      └─ proposal
|
|      |      └─ model
|
|      |      └─ Proposal.class
|
|      └─ vendor
|
|      |      └─ model
|
|      |      └─ Vendor.class
|
|      |      └─ Assignment.class
|
|      |      └─ Quotation.class
|
|      └─ grievance
|
|      |      └─ model
|
|      |      └─ Grievance.class
|
|      └─ system
|
|      └─ CivicPulse.class
|
|      └─ CivicPulseGUI.class
└─ lib

```

- | └─ (empty or optional JDBC driver, e.g., mysql-connector-java.jar if using MySQL)
- | └─ resources
- |   └─ database
- |    └─ schema.sql
- |    └─ config
- |      └─ dbconfig.properties
- | └─ docs
- |   └─ design.md
- |    └─ user\_manual.md
- | └─ scripts
- |   └─ compile.sh
- |    └─ run.sh
- | └─ README.md
- └─ LICENSE

## Detailed Breakdown of Directories and Files

### 1. src Directory

- **Purpose:** Contains all Java source files, organized by package.
- **Structure:** Mirrors the package hierarchy (com.civimpulse).
- **Files:**

- `src/com/civimpulse/core/Category.java`: Enum for categories (e.g., WATER, ELECTRICITY).
- `src/com/civimpulse/core/Role.java`: Enum for user roles (e.g., CITIZEN, AUTHORITY, VENDOR).
- `src/com/civimpulse/user/model/User.java`: Abstract base class for users.
- `src/com/civimpulse/user/model/Citizen.java`: Model for citizen users.
- `src/com/civimpulse/user/model/Authority.java`: Model for authority users.
- `src/com/civimpulse/user/model/VendorUser.java`: Model for vendor users.
- `src/com/civimpulse/issue/model/Issue.java`: Model for issues.
- `src/com/civimpulse/issue/model/Comment.java`: Model for comments on issues or proposals.
- `src/com/civimpulse/proposal/model/Proposal.java`: Model for proposals.
- `src/com/civimpulse/vendor/model/Vendor.java`: Model for vendors.
- `src/com/civimpulse/vendor/model/Assignment.java`: Model for vendor assignments.
- `src/com/civimpulse/vendor/model/Quotation.java`: Model for quotations.
- `src/com/civimpulse/grievance/model/Grievance.java`: Model for grievances.
- `src/com/civimpulse/system/CivicPulse.java`: Main application logic (e.g., database connection).
- `src/com/civimpulse/system/CivicPulseGUI.java`: GUI implementation (already provided).

## 2. bin Directory

- **Purpose:** Contains compiled .class files, generated after compiling the source files.
- **Structure:** Mirrors the src directory structure.
- **Files:** Each .java file in src has a corresponding .class file in bin after compilation (e.g., Category.class, CivicPulseGUI.class).

### 3. lib Directory

- **Purpose:** Stores external libraries (e.g., JDBC drivers).
- **Files:**
  - (Optional) mysql-connector-java.jar: If using MySQL, this would be the JDBC driver. For this project, it's assumed to use a database like MySQL, but the driver isn't strictly required if using an in-memory database like H2 for testing.

### 4. resources Directory

- **Purpose:** Contains non-code resources like database scripts and configuration files.
- **Subdirectories:**
  - resources/database/
    - schema.sql: SQL script to create the database tables (citizens, authorities, vendors, issues, proposals, grievances, assignments, quotations, comments).
  - resources/config/
    - dbconfig.properties: Configuration file for database connection (e.g., URL, username, password).

### 5. docs Directory



- **Purpose:** Contains documentation for the project.
- **Files:**
  - design.md: High-level design document outlining the architecture.
  - user\_manual.md: Instructions for using the application.

## 6. scripts Directory

- **Purpose:** Contains shell scripts for compiling and running the project (assuming a simple setup without build tools).
- **Files:**
  - compile.sh: Script to compile all Java files.
  - run.sh: Script to run the application.

## 7. README.md

- **Purpose:** Provides an overview of the project, setup instructions, and usage.

## 8. LICENSE

- **Purpose:** Specifies the licensing terms (e.g., MIT License).

Database :-

-- Create the civicpulse\_db database

```
CREATE DATABASE IF NOT EXISTS civicpulse_db;
```

```
USE civicpulse_db;
```

```
-- Drop existing tables to ensure a clean setup (optional, comment out if you want to preserve data)
```

```
-- Table for citizens
```

```
CREATE TABLE citizens (  
  
    id INT AUTO_INCREMENT PRIMARY KEY,  
  
    name VARCHAR(100) NOT NULL,  
  
    gender VARCHAR(10),  
  
    age INT,  
  
    email VARCHAR(100),  
  
    aadhar_no VARCHAR(12),  
  
    mobile_no VARCHAR(15),  
  
    state VARCHAR(100),  
  
    district VARCHAR(100),  
  
    local_area VARCHAR(100),  
  
    username VARCHAR(50) NOT NULL UNIQUE,  
  
    password VARCHAR(50) NOT NULL  
  
);
```

```
-- Table for authorities
```

```
CREATE TABLE authorities (
```

```
id INT AUTO_INCREMENT PRIMARY KEY,  
  
name VARCHAR(100) NOT NULL,  
  
username VARCHAR(50) NOT NULL UNIQUE,  
  
password VARCHAR(50) NOT NULL  
  
);
```

-- Table for vendors

```
CREATE TABLE vendors (  
  
id INT AUTO_INCREMENT PRIMARY KEY,  
  
name VARCHAR(100) NOT NULL,  
  
email VARCHAR(100),  
  
mobile_no VARCHAR(15),  
  
expertise VARCHAR(50),  
  
username VARCHAR(50) NOT NULL UNIQUE,  
  
password VARCHAR(50) NOT NULL  
  
);
```

-- Table for issues

```
CREATE TABLE issues (  
  
issue_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
description TEXT NOT NULL,  
  
state VARCHAR(100) NOT NULL,  
  
district VARCHAR(100) NOT NULL,  
  
locality VARCHAR(100),  
  
citizen_id INT NOT NULL,  
  
category VARCHAR(50) NOT NULL,  
  
status VARCHAR(50) NOT NULL,  
  
votes INT DEFAULT 0,  
  
reported_date DATE NOT NULL,  
  
FOREIGN KEY (citizen_id) REFERENCES citizens(id)  
  
);
```

-- Table for proposals

```
CREATE TABLE proposals (  
  
    proposal_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    description TEXT NOT NULL,  
  
    state VARCHAR(100) NOT NULL,  
  
    district VARCHAR(100) NOT NULL,  
  
    locality VARCHAR(100),  
  
    citizen_id INT NOT NULL,
```

```
status VARCHAR(50) NOT NULL,  
  
votes INT DEFAULT 0,  
  
submitted_date DATE NOT NULL,  
  
FOREIGN KEY (citizen_id) REFERENCES citizens(id)  
  
);  
  
-- Table for grievances  
  
CREATE TABLE grievances (  
  
    grievance_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    description TEXT NOT NULL,  
  
    issue_id INT NOT NULL,  
  
    citizen_id INT NOT NULL,  
  
    filed_date DATE NOT NULL,  
  
    status VARCHAR(50) NOT NULL,  
  
    FOREIGN KEY (issue_id) REFERENCES issues(issue_id),  
  
    FOREIGN KEY (citizen_id) REFERENCES citizens(id)  
  
);  
  
-- Table for assignments  
  
CREATE TABLE assignments (
```

```
assignment_id VARCHAR(50) PRIMARY KEY,  
  
issue_id INT NOT NULL,  
  
vendor_id INT NOT NULL,  
  
assigned_date DATE NOT NULL,  
  
status VARCHAR(50) NOT NULL,  
  
FOREIGN KEY (issue_id) REFERENCES issues(issue_id),  
  
FOREIGN KEY (vendor_id) REFERENCES vendors(id)  
  
);
```

-- Table for quotations

```
CREATE TABLE quotations (  
  
    quotation_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    issue_id INT NOT NULL,  
  
    vendor_id INT NOT NULL,  
  
    description TEXT NOT NULL,  
  
    price DECIMAL(10, 2) NOT NULL,  
  
    estimated_days INT NOT NULL,  
  
    details TEXT NOT NULL,  
  
    submission_date DATE NOT NULL,  
  
    status VARCHAR(50) NOT NULL,
```

```
FOREIGN KEY (issue_id) REFERENCES issues(issue_id),

FOREIGN KEY (vendor_id) REFERENCES vendors(id)

);

-- Table for comments

CREATE TABLE comments (

    comment_id INT AUTO_INCREMENT PRIMARY KEY,

    citizen_id INT NOT NULL,

    issue_id INT,

    proposal_id INT,

    comment_text TEXT NOT NULL,

    comment_date DATE NOT NULL,

    FOREIGN KEY (citizen_id) REFERENCES citizens(id),

    FOREIGN KEY (issue_id) REFERENCES issues(issue_id),

    FOREIGN KEY (proposal_id) REFERENCES proposals(proposal_id),

    CONSTRAINT chk_comment_target CHECK (

        (issue_id IS NOT NULL AND proposal_id IS NULL) OR

        (issue_id IS NULL AND proposal_id IS NOT NULL)

    )

);
```

-- Insert a default authority for testing

```
INSERT INTO authorities (name, username, password) VALUES ('Admin', 'admincivicpulse',  
'admin123');
```

-- Insert a default citizen for testing

```
INSERT INTO citizens (name, gender, age, email, aadhar_no, mobile_no, state, district,  
local_area, username, password)  
  
VALUES ('John Doe', 'Male', 30, 'john.doe@example.com', '123456789012', '9876543210',  
'Rajasthan', 'Kota', 'Downtown', 'johndoe', 'pass123');
```

-- Insert a default vendor for testing

```
INSERT INTO vendors (name, email, mobile_no, expertise, username, password)  
  
VALUES ('Vendor One', 'vendor1@example.com', '9123456789', 'CONSTRUCTION', 'vendor1',  
'vendorpass');
```

```
USE civicpulse_db;
```

```
SELECT * FROM citizens;
```

```
SELECT * FROM authorities;
```

```
SELECT * FROM vendors;
```

```
SELECT * FROM issues;
```

```
SELECT * FROM proposals;
```



```
SELECT * FROM comments;
```

```
SELECT * FROM assignments;
```

```
SELECT * FROM quotations; -- Fixed typo: quotationsvendors → quotations
```

```
SELECT * FROM grievances;
```

```
-- Exit MySQL--
```

```
vv
```

## Dependencies Between Packages

- **com.civicpulse.core:** Used by com.civicpulse.user, com.civicpulse.issue, com.civicpulse.vendor, com.civicpulse.notification, com.civicpulse.quotation.
- **com.civicpulse.user:** Depends on com.civicpulse.core, com.civicpulse.issue, com.civicpulse.grievance, com.civicpulse.notification, com.civicpulse.persistence, com.civicpulse.util.
- **com.civicpulse.issue:** Depends on com.civicpulse.user, com.civicpulse.quotation, com.civicpulse.persistence, com.civicpulse.util.
- **com.civicpulse.vendor:** Depends on com.civicpulse.core, com.civicpulse.issue, com.civicpulse.assignment, com.civicpulse.quotation, com.civicpulse.persistence, com.civicpulse.util.
- **com.civicpulse.authority:** Depends on com.civicpulse.user, com.civicpulse.issue, com.civicpulse.vendor, com.civicpulse.assignment, com.civicpulse.quotation, com.civicpulse.grievance, com.civicpulse.reporting, com.civicpulse.util.
- **com.civicpulse.grievance:** Depends on com.civicpulse.user, com.civicpulse.issue, com.civicpulse.persistence, com.civicpulse.util.

- **com.civicpulse.notification:** Depends on com.civicpulse.user, com.civicpulse.persistence, com.civicpulse.util.
  - **com.civicpulse.assignment:** Depends on com.civicpulse.issue, com.civicpulse.vendor, com.civicpulse.persistence, com.civicpulse.util.
  - **com.civicpulse.quotation:** Depends on com.civicpulse.issue, com.civicpulse.vendor, com.civicpulse.notification, com.civicpulse.persistence, com.civicpulse.util.
  - **com.civicpulse.persistence:** Depends on com.civicpulse.util.
  - **com.civicpulse.reporting:** Depends on com.civicpulse.issue, com.civicpulse.vendor, com.civicpulse.grievance, com.civicpulse.persistence, com.civicpulse.util.
  - **com.civicpulse.util:** Used by all packages for validation, exceptions, and logging.
  - **com.civicpulse.audit:** Depends on com.civicpulse.assignment, com.civicpulse.vendor, com.civicpulse.system, com.civicpulse.util.
  - **com.civicpulse.system:** Depends on all other packages to orchestrate the system.
- 

## Justification of Package Structure

### 1. Modularity:

- Each package focuses on a specific concern, such as com.civicpulse.issue for issue management and com.civicpulse.quotation for quotation handling.
- Sub-packages like model, service, and exception further modularize logic for easier maintenance and testing.

### 2. Separation of Phases:

- Phase 1 (citizen operations) is covered by com.civicpulse.issue, com.civicpulse.user, and com.civicpulse.grievance.
- Phase 2 (vendor/authority operations) is covered by com.civicpulse.vendor, com.civicpulse.authority, com.civicpulse.assignment, and com.civicpulse.quotation.
- Shared functionality, such as notifications and persistence, is in com.civicpulse.notification and com.civicpulse.persistence.

### 3. Reusability:

- com.civicpulse.util provides reusable validation, exception handling, and logging utilities.

- `com.civimpulse.core` centralizes shared enums and interfaces for system-wide reuse.
- `com.civimpulse.notification` and `com.civimpulse.persistence` are used across both phases.

#### 4. **Scalability:**

- Sub-packages allow for future expansion, such as adding new models in `com.civimpulse.vendor.model`.
- Collections like `ArrayList`, `HashMap`, `LinkedList`, and `PriorityQueue` support large-scale data management.

#### 5. **Maintainability:**

- Clear package names, such as `com.civimpulse.reporting`, make it easy to locate functionality.
- Separation of concerns reduces coupling between modules.
- Dedicated exception sub-packages improve error handling organization.

#### 6. **OOP Compliance:**

- Uses inheritance, such as `User` → `Authority` and `Vendor` → `InfrastructureVendor`.
- Implements abstraction through interfaces (`Manageable`, `Reportable`) and abstract classes (`User`).
- Encapsulates data with private attributes and public methods.

Supports polymorphism with overloaded methods and overridden behavior in subclasses.

## Conclusion

The **CivicPulse project** successfully developed a robust citizen engagement portal that bridges the gap between citizens, authorities, and vendors, fostering transparent and efficient governance. Designed as a **Java-based application**, the project leveraged **Java Swing** for an intuitive graphical user interface, consolidated into a single CivicPulseGUI.java file using CardLayout to manage multiple panels such as **Main Menu**, **Citizen Login/Register/Dashboard**, **Authority Login/Dashboard**, and **Vendor Login/Register/Dashboard**. The system integrates seamlessly with a **MySQL database** (civicpulse\_db) to handle data persistence, enabling functionalities like citizen registration, issue reporting, proposal submission, grievance filing, vendor assignment, and quotation submission.

Throughout the development process, challenges such as **database connectivity issues**, **GUI navigation**, and **file organization** were addressed by refining the project structure, ensuring proper package management (e.g., com.civicpulse.system, com.civicpulse.core), and resolving compilation errors (e.g., renaming CiviPulse.java to CivicPulse.java, adding missing classes like AuthorityLoginFrame). The final implementation ensures all buttons are functional, allowing citizens to report issues (e.g., potholes categorized under **INFRASTRUCTURE**), authorities to assign vendors, and vendors to submit quotations. Placeholders for future enhancements, such as voting and commenting, have been strategically added.

Stored in C:\Myjava\CivicPulseApp\src, the project compiles and runs smoothly using commands like javac and java with the MySQL Connector JAR, providing a scalable foundation for future features such as **advanced reporting** or **user analytics**. Ultimately, the project empowers communities to voice concerns and collaborate on solutions effectively.

## References

The development of the CivicPulse project relied on several key resources and tools to achieve its objectives:

- **Programming Language:** Java 17, with Java Swing for the graphical user interface, as documented in the [Java Swing Tutorial by Oracle](#).
- **Database:** MySQL Community Server 9.3.0, managed using the MySQL Connector/J library (mysql-connector-java-9.3.0.jar) for database connectivity, as referenced in the [MySQL documentation](#).
- **Project Structure:** Best practices outlined in the [Java Package Naming Conventions](#) informed package management.
- **GUI Management:** CardLayout for multiple panels, based on examples in the [Java Tutorials on Layout Managers](#).
- **Database Operations:** Secure queries via prepared statements followed guidelines from the [JDBC Basics tutorial](#).
- **Troubleshooting:** Community discussions on Stack Overflow (e.g., resolving "Access denied" errors: [link](#)).
- **GUI Design Inspiration:** Adapted from "Lab Worksheet 11 GUI v1 (2).docx" by Vidyashilp University, introducing Swing components and multi-frame navigation concepts.

## OUTPUTS

 CivicPulse - Citizen Engagement Portal

### Citizen Registration


Name:	<input type="text" value="Pruthviraj Shinde"/>
Gender:	<input type="text" value="Male"/>
Age:	<input type="text" value="23"/>
Email:	<input type="text" value="pruthvi.raj@gmail.com"/>
Aadhar No.:	<input type="text" value="567834567809"/>
Mobile No.:	<input type="text" value="9765437689"/>
State:	<input type="text" value="Maharashtra"/>
District:	<input type="text" value="Nagpur"/>
Local Area:	<input type="text" value="Saheed Road"/>
Username:	<input type="text" value="pruthvi01"/>
Password:	<input type="password" value="...."/>

Submit Back

Message

 Registration Successful for Pruthviraj Shinde

OK


 CivicPulse - Citizen Engagement Portal


### Citizen Login

**Username:**

**Password:**

Message ×

 **Welcome Pruthviraj Shinde**

 CivicPulse - Citizen Engagement Portal

### Citizen Dashboard

Report Issue

Submit Proposal

Vote

Comment

File Grievance

View Dashboard

Logout





## Report Issue

**Description:**

**State:**

**District:**

**Locality:**

**Category:**

Submit

Back

Message



Issue reported successfully.

OK



## Submit Proposal

Description:

Install Solar Panel

State:

Haryana

District:

Hisar

Locality:

Mill Gate

Submit

Back

Message



Proposal submitted successfully.

OK



## Vote on Issue/Proposal

Vote On:

Issue

ID:

2

Vote

Back

Message



Voted on issue successfully.

OK



### Comment on Issue/Proposal

Comment On:

Issue

ID:

4

Comment:

Need to be Repair asap

Submit

Back

Message



Comment added to issue successfully.

OK



## File Grievance

Description:

Frequent Power Cut

Issue ID:

4

Submit

Back

Message



Grievance filed successfully.

OK



## Review Issues

===== Issues List =====

Issue ID: 1, Description: Water Drainage, Citizen ID: 1, Status: ASSIGNED, Votes: 1

Issue ID: 2, Description: Broken Water Pipeline, Citizen ID: 1, Status: REPORTED, Votes: 0

Issue ID: 3, Description: Potholes on Road, Citizen ID: 1, Status: ASSIGNED, Votes: 0

Issue ID: 4, Description: Frequent Power Cut, Citizen ID: 2, Status: REPORTED, Votes: 0

[Back](#)



## Review Proposals

===== Proposals List =====

Proposal ID: 1, Description: Build a park, Citizen ID: 1, Status: SUBMITTED, Votes: 1


Proposal ID: 2, Description: Build a Library, Citizen ID: 1, Status: SUBMITTED, Votes: 0

Proposal ID: 3, Description: Build a Play School, Citizen ID: 1, Status: SUBMITTED, Votes: 0

Proposal ID: 4, Description: Install Solar Panel, Citizen ID: 3, Status: SUBMITTED, Votes: 0

|

[Back](#)

 CivicPulse - Citizen Engagement Portal

## Your Dashboard

===== Your Dashboard =====

**Reported Issues:**  
Issue ID: 5, Description: Supply Water Cut, Status: REPORTED, Votes: 0


**Submitted Proposals:**  
Proposal ID: 4, Description: Install Solar Panel, Status: SUBMITTED, Votes: 0

**Filed Grievances:**  
Grievance ID: 2, Issue ID: 4, Description: Frequent Power Cut, Status: FILED

**Your Comments:**  
Comment ID: 3, Issue ID: 4, Comment: Need to be Repair asap, Date: 2025-05-05

Back




 CivicPulse - Citizen Engagement Portal

## Authority Login

**Username:**

**Password:**

Message ×

 **Welcome Admin**



## View Citizens


===== Citizens List =====

Citizen ID: 1, Name: John Doe, Username: johndoe, Email: john.doe@example.com

Citizen ID: 2, Name: Ayush Kumar, Username: ayush01, Email: ayush@gmail.com

Citizen ID: 3, Name: Pruthviraj Shinde, Username: pruthvi01, Email: pruthvi.raj@gmail.com

Back

 CivicPulse - Citizen Engagement Portal

## Vendor Login

**Username:**

sumit01

**Password:**

....

Login

Back

Message



Welcome Sumit Kr

OK



### Submit Quotation

Issue ID:

Description:

Price:

Estimated Days:

Details:

Submit

Back

Message



Quotation submitted successfully.

OK