TED UNIVERSITY

CMPE 313/SENG 214

Software Engineering

Citizen Portal Platform

Software Design Document (SDD)

Section 3 – Team 2

15.05.2024

Team Members: Mustafa Pınarcı, Ege İzmir, Bartu Özen, Ahmet Tokgöz, Onat Keser, Emir Can Tokalakoğlu

Revision History

Name	Date	Reason For Changes	Version

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

The goal of this software design document (SDD) is to provide an overview of the system design and architectural plan for the creation of a Citizen Portal that will promote public participation in the governing structure of our municipality. The aim of this project is to offer a detailed blueprint for developing a digital platform that enables citizens to actively participate in decision-making. Enabling residents to participate actively through voting on ideas, recommending new discussion subjects, providing comments, and safely managing their accounts is the main objective. This agreement also attempts to guarantee that the portal preserves user privacy and promotes a secure, welcoming atmosphere for civic participation. Developers, project managers, stakeholders, and any other parties participating in the software development lifecycle are the target audience for this SDD.

1.2 Scope

The creation of a Citizen Portal, which acts as a primary location for public involvement in our municipality's government, is included in the project's scope. Through the portal, locals will be able to vote on proposals made by the city council, suggest new topics for discussion, feedback on submissions made by other people, and manage their own profiles. The Citizen Portal will use user authentication technologies to confirm users' identities as citizens of the city while protecting their anonymity on the network. The objectives of this project are to increase transparency in governance and enhance civic engagement. Better accessibility to decision-making procedures, increased openness, and the advancement of a more inclusive and participatory democracy are among the advantages of the Citizen Portal.

1.3 Overview

This paper aims to provide a comprehensive overview of the architecture and software design for the Citizen Portal project. It covers sections on the project's introduction, system overview, system architecture, data design, component design, human interface design, and requirements matrix. Each section details various aspects of the Citizen Portal project, including its purpose, scope, architectural design, data structures, component functionalities, user interface elements, and alignment with functional requirements. The document aims to provide a comprehensive understanding of the project's objectives, specifications, and deliverables, serving as a guide for the development team.

2. System Overview

The Citizen Portal is a complete digital ecosystem designed to encourage active participation in civic life and participatory governance in the local government. It functions as an active medium through which citizens, members of the city council, and administrators can jointly negotiate and shape the terrain of decision-making.

Functionality:

- **Voting on Proposals:** The portal furnishes residents with the capacity to engage directly in the democratic process by reviewing and voting on proposals deliberated within the city council.

Through an intuitive interface, citizens can access relevant information, evaluate proposals, and cast informed votes, thereby democratizing decision-making and amplifying community voices.

- **Topic Submission:** Residents are empowered to instigate discussions on pertinent civic matters by proposing new topics for consideration within the portal. This functionality fosters inclusivity and responsiveness, enabling diverse perspectives to inform the municipal agenda and catalyze proactive dialogue on emergent issues.
- **Feedback Mechanism:** Anchored in principles of transparency and dialogue, the portal facilitates reciprocal communication between residents and municipal stakeholders. Citizens can offer constructive feedback on proposals submitted by peers, engendering a culture of collaboration and iterative refinement within the decision-making process.
- **Profile Management:** Recognizing the importance of identity verification and user autonomy, the portal affords residents granular control over their personal profiles. From updating contact information and managing communication preferences, users can curate their digital personas, fostering a sense of ownership and trust within the platform.

Context:

The project's context involves the transition from traditional voting methods to a digital platform, emphasizing citizen engagement and transparency in municipal governance. Within this context, the Citizen Portal Platform (CPP) aims to empower citizens to actively participate in decision-making processes by providing accessible interfaces for proposal submission, feedback provision, and voting. Additionally, the platform facilitates collaboration between citizens, city workers, and the mayor, fostering a transparent ecosystem where proposals and ongoing processes are readily accessible and comprehensible to all stakeholders.

Design:

In essence, the portal embodies a combination of usability, accessibility, and security, supported by strong architectural frameworks and user-centered design principles. The Citizen Portal provides users access to features in the system and emphasizes the importance of user-friendly navigation, responsive design, and strict data privacy measures. Through feedback loops with stakeholders and iterative development, the site adapts naturally to suit the changing needs of its wide range of users.

3. System Architecture

3.1 Architectural Design:

The Citizen Portal system's architectural design refers to a modular program structure designed to effectively manage the system's complexity and provide scalability, maintainability, and extension. The architecture identifies the following primary subsystems:

1. **User Interface Subsystem:** In charge of introducing the municipal council members, administrators, and residents to the user interface. It manages user interactions and exchanges data with other subsystems in order to get and provide necessary information.

2. **Authentication Subsystem:** Manages the processes for user permission and authentication. It keeps track of user session data, confirms user credentials, and guarantees safe system access.

3. **Proposal Management Subsystem:** Helps with the administration of suggestions made by residents and members of the city council. In addition to ensuring the integrity and security of proposal data, it enables users to create, view, vote on, and comment on proposals.

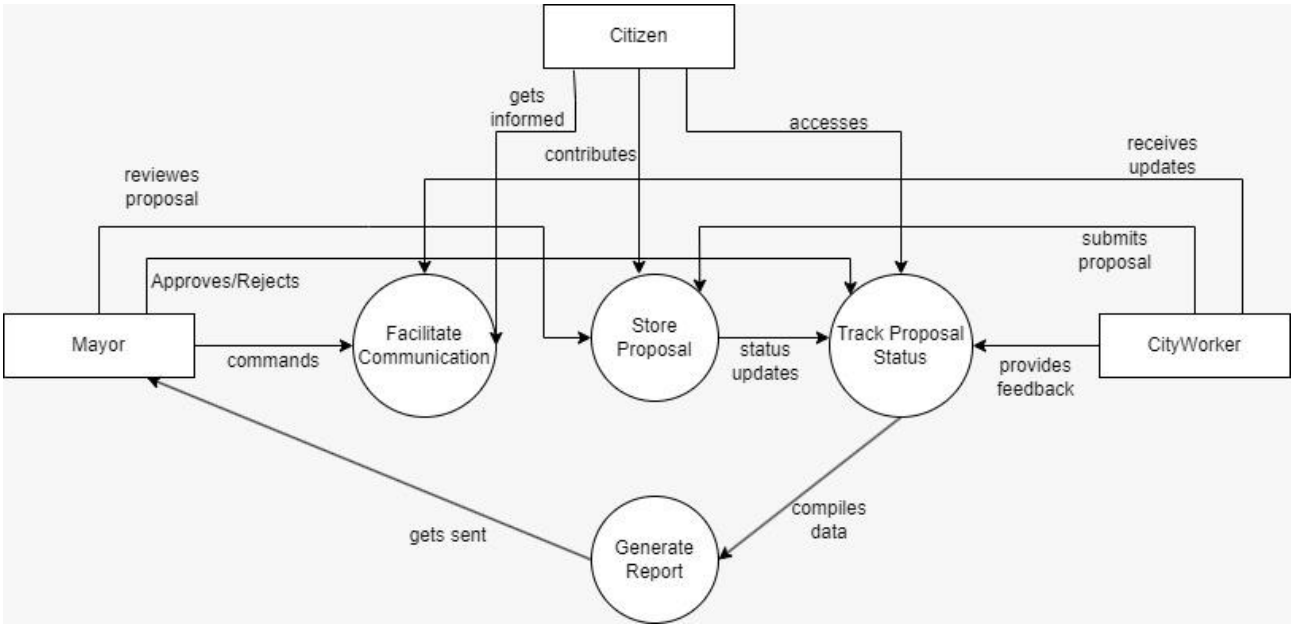
4. **Feedback Management Subsystem:** Handles the collection and management of feedback provided by users on proposals and other system functionalities. It enables users to submit feedback, view feedback provided by others, and interact with feedback data.

5. **Database Subsystem:** Acts as the central data repository for the system, storing user profiles, proposal data, feedback data, and other system-related information. It ensures data integrity, availability, and security.

These subsystems collaborate with each other to achieve the desired functionality of the Citizen Portal system. For example, the User Interface Subsystem interacts with the Authentication Subsystem to authenticate users before providing access to system functionalities. Similarly, the Proposal Management Subsystem interacts with the Feedback Management Subsystem to allow users to provide feedback on proposals. The Database Subsystem serves as the backbone of the system, providing data storage and retrieval services to all other subsystems.

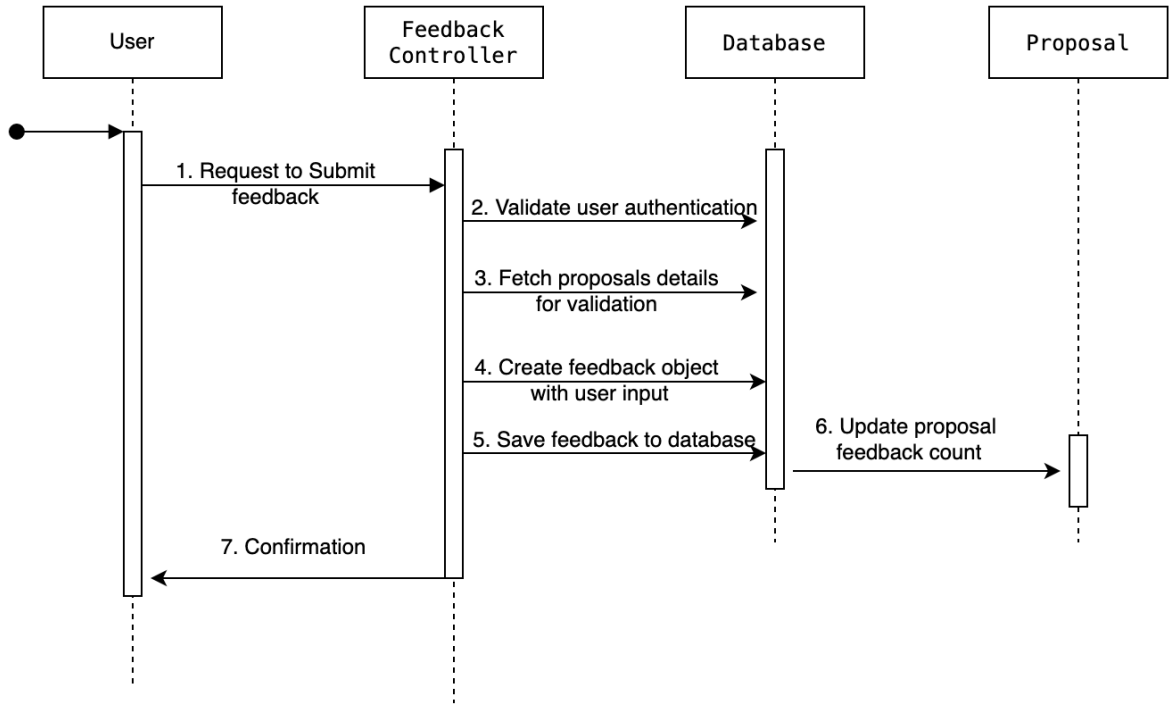
3.2 Decomposition Description:

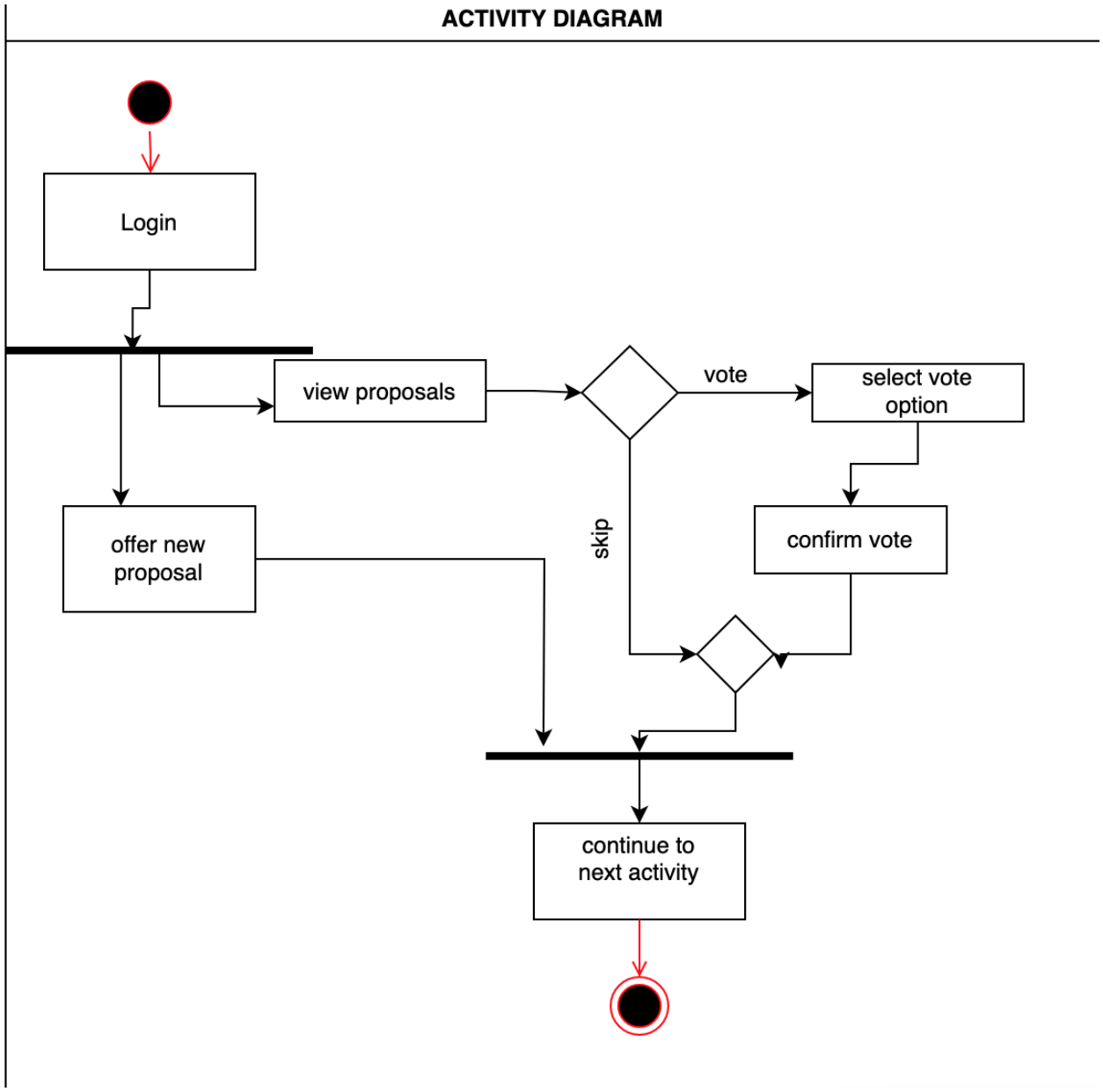
The decomposition of the subsystems in the architectural design can be represented using various diagrams and models. For a functional description, top-level data flow diagrams (DFDs) and structural decomposition diagrams can be used to illustrate the flow of data and the breakdown of subsystems into smaller components. For an object-oriented description, subsystem models, object diagrams, generalization hierarchy diagrams, aggregation hierarchy diagrams, UML package diagrams, interface specifications, and sequence diagrams can be included to provide a detailed view of the system's structure and behavior.



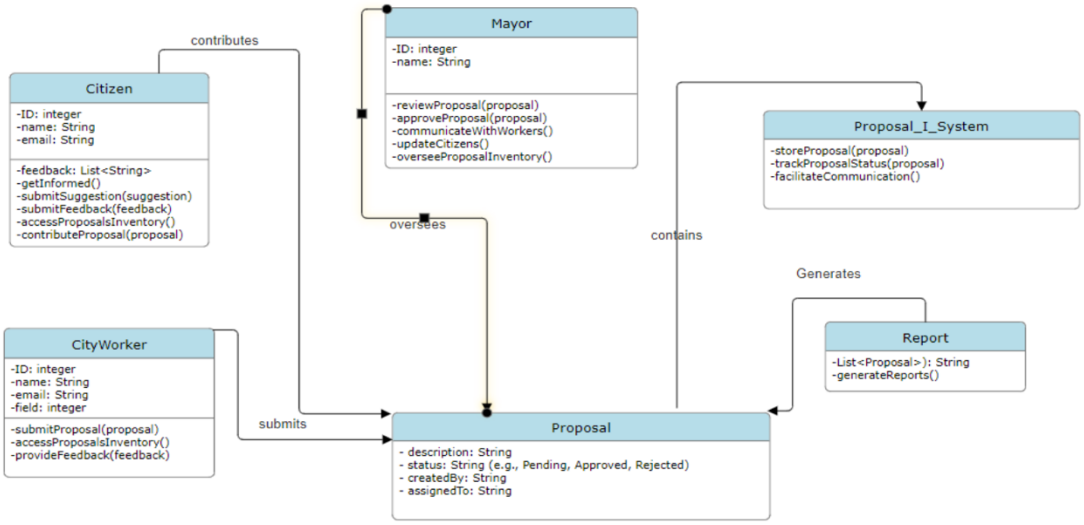
3.3 Design Rationale:

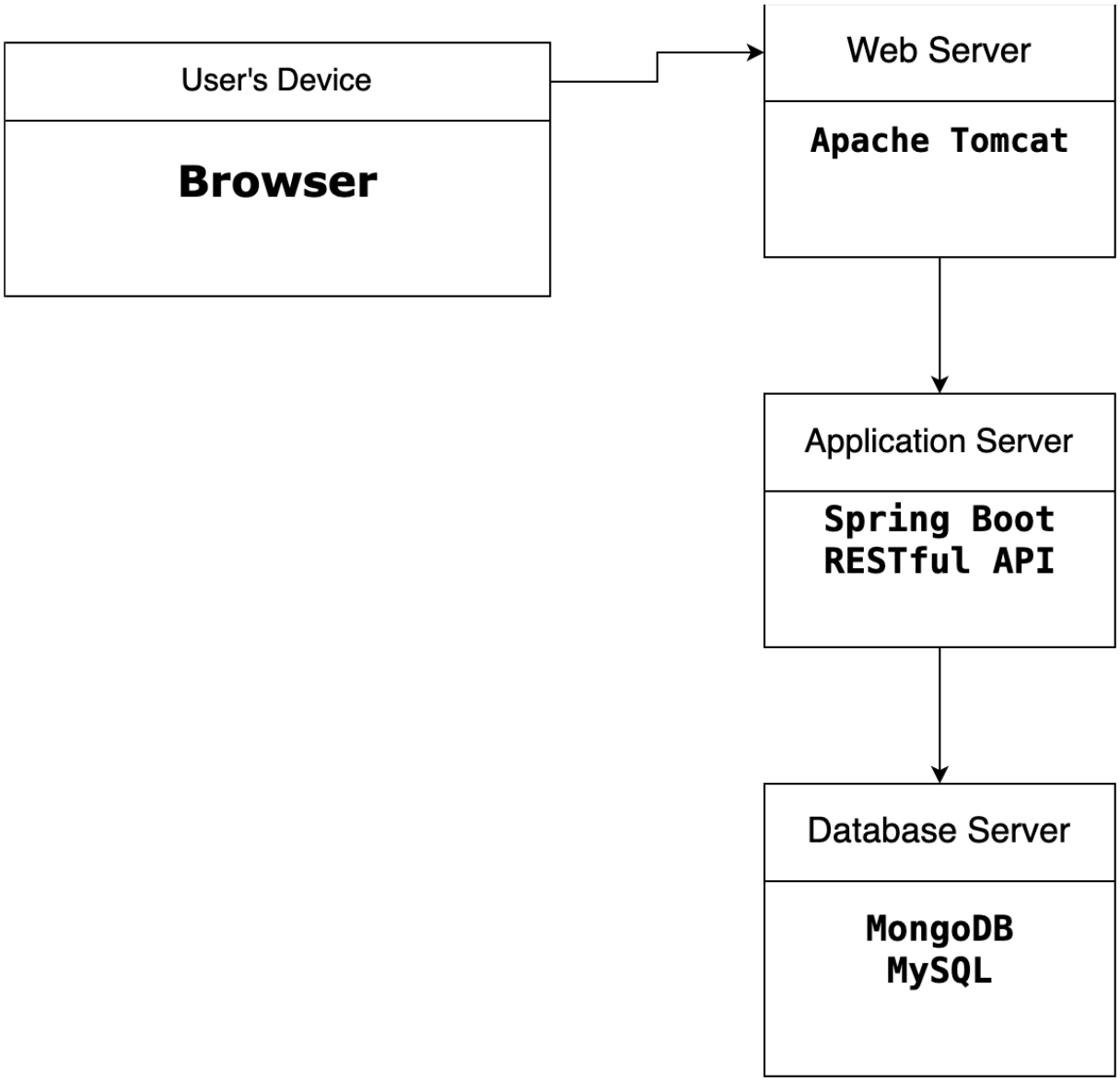
The selected architecture for the Citizen Portal system was chosen based on several factors, including scalability, maintainability, extensibility, and security. A modular program structure was adopted to partition the system's responsibilities into manageable subsystems, allowing for easier development, testing, and maintenance of the system. Critical issues such as user authentication, data integrity, and system performance were carefully considered during the design process. Other architectures, such as monolithic or microservices-based architectures, were evaluated but not chosen due to their limitations in terms of flexibility, scalability, or complexity.

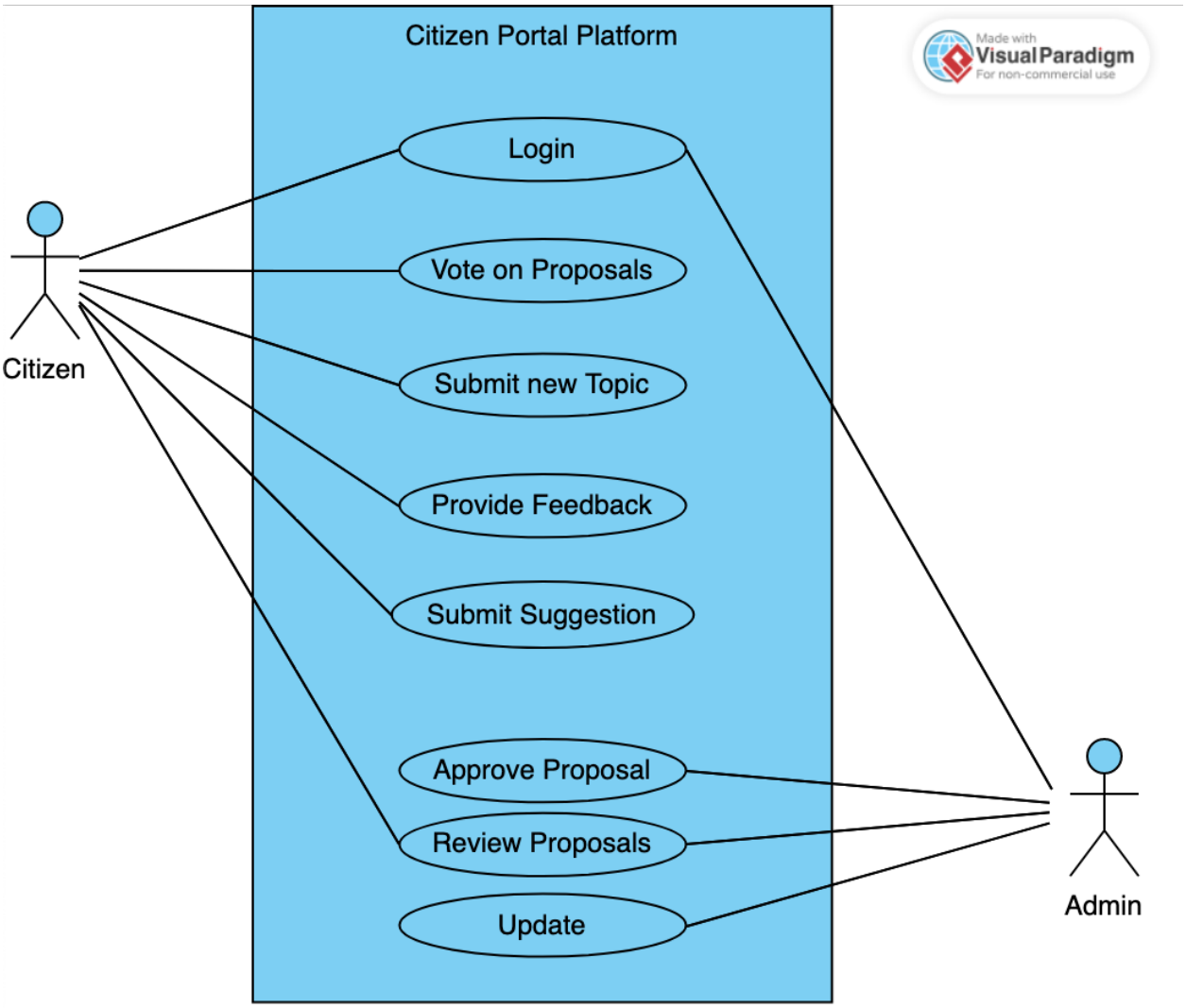

User
Feedback Controller
Database
Proposal
1. Request to Submit feedback
2. Validate user authentication
3. Fetch proposals details for validation
4. Create feedback object with user input
5. Save feedback to database
6. Update proposal feedback count
7. Confirmation

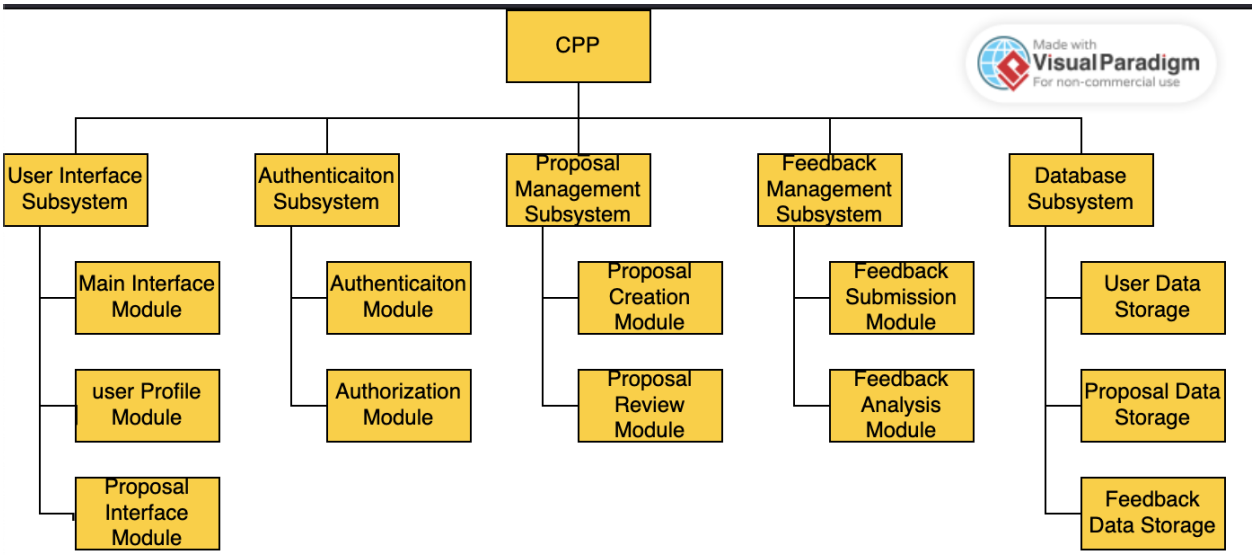

ACTIVITY DIAGRAM
Login
view proposals
vote
select vote option
offer new proposal
skip
confirm vote
continue to next activity

Class Diagram




User's Device
Browser
Web Server
Apache Tomcat
Application Server
Spring Boot
RESTful API
Database Server
MongoDB
MySQL


Citizen Portal Platform
Made with Visual Paradigm For non-commercial use
Login
Vote on Proposals
Submit new Topic
Provide Feedback
Submit Suggestion
Approve Proposal
Review Proposals
Update
Citizen
Admin


CPP
User Interface Subsystem
Authenticaiton Subsystem
Proposal Management Subsystem
Feedback Management Subsystem
Database Subsystem
Main Interface Module
user Profile Module
Proposal Interface Module
Authenticaiton Module
Authorization Module
Proposal Creation Module
Proposal Review Module
Feedback Submission Module
Feedback Analysis Module
User Data Storage
Proposal Data Storage
Feedback Data Storage
Made with Visual Paradigm
For non-commercial use

4. Data Design

4.1 Data Description

This section displays how the information domain of the Citizen Portal Platform is transformed into various data structures defining the storage, processing, and organization of key system entities.

- **Data Entities and Structures:** The system is built around several data entities which are User, Proposal, Vote, and Feedback. These entities capture all the attributes to support the functionalities of the Citizen Portal Platform:

- **User:** Stores information about the users of the system such as citizens, city workers, and administrators. Attributes include user ID, name, role, and contact details.

- **Proposal:** Contains details about the proposals submitted by users. Attributes include proposal ID, title, description, status, submission date, and author ID.

- **Vote:** Records the votes cast by users on different proposals. Attributes include vote ID, proposal ID, user ID, and vote type (approve, reject, abstain).

- **Feedback:** Stores feedback submitted by users on proposals. Attributes include feedback ID, content, author ID, and associated proposal ID.

- **Data Storage:** The system utilizes a PostgreSQL database for storing all data.

- **Data Processing:** Data is processed using a combination of SQL queries and application logic implemented in the system's backend. To enhance data management, an Object-Relational Mapping (ORM) tool is employed. This approach simplifies the development process, ensures consistency between the database schema and the application model, and aids in maintaining data integrity and transaction management. This method helps to maintain data integrity and transaction management, makes the development process simpler, and guarantees consistency between the application model and the database schema.

- **Organization:** A relational schema is used to organize data, allowing effective data retrieval and manipulation. Foreign keys and indexing are used to preserve relationships between entities in order to improve performance. The complexity of database operations is greatly reduced by the use of ORM, which enables the smooth integration and management of these relationships.

This structured approach to data handling ensures that the Citizen Portal Platform can efficiently manage the various interactions and functionalities required by its users, while maintaining data integrity and supporting scalability.

4.2 Data Dictionary

Citizen information

Attribute	Type	Size
ID	Integer	12
Name	String	50
Email	String	50

Cityworker information

Attribute	Type	Size
ID	Integer	12
Name	String	50
Email	String	50
Field	Integer	10

Report

Attribute	Type	Size
Proposal List	List of Strings	#

Mayor Information

Attribute	Type	Size
ID	Integer	12
Name	String	50

Proposal Information

Attribute	Type	Size
Description	String	100
Status	String	50
created by	String	50
AssignedTo	String	50

In this section, we describe how the major data or system entities are stored, processed, and organized within our Citizen Portal Platform system using PostgreSQL as the database system.

1. Proposal List

   - Description: This entity represents a list of proposals submitted by citizens or city workers.
   - Storage: Stored as a PostgreSQL table with a column for each proposal, allowing for easy retrieval and management.
   - Processing: Accessed during proposal submission, viewing, and management processes.
   - Organization: Organized within the table based on proposal submission time or other relevant criteria.

2. Citizen Information

   - Description: Stores information about citizens using the platform.
   - Storage: Attributes such as ID, Size, Email, and Field are stored in a PostgreSQL table, ensuring structured data storage.

- Processing: Used for user authentication, profile management, and communication purposes.

- Organization: Structured within the table using unique IDs for efficient retrieval and updating.

3. City Worker Information

   - Description: Holds data related to city workers registered on the platform.

   - Storage: Similar to Citizen Information, storing ID, Size, and Email for city workers in a PostgreSQL table.

   - Processing: Used for authentication, task assignment, and communication with city workers.

   - Organization: Structured within the table based on unique IDs for streamlined access and management.

4. Report

   - Description: Contains reports submitted through the platform.

   - Storage: Attributes like ID, Size, Email, and report content are stored in a PostgreSQL table.

   - Processing: Accessed during report submission, viewing, and processing by authorized personnel.

   - Organization: Arranged within the table by submission time or other relevant criteria for tracking purposes.

5. Mayor Information

   - Description: Stores details about the mayor, including ID and Name.

   - Storage: Mayor's ID and Name are stored in a PostgreSQL table for identification and reference purposes.

   - Processing: Used for authentication and access control for mayor-related functionalities.

- Organization: Single entry storage within the table, reflecting the unique nature of the mayor's role.

6. Proposal Information

    - Description: Stores detailed information about each proposal submitted.
    - Storage: Includes attributes such as ID, Description, Status, Created by, and AssignedTo in a PostgreSQL table.
    - Processing: Used for proposal management, status tracking, and assignment handling.
    - Organization: Organized within the table using unique IDs for efficient retrieval and tracking of proposal details.

Data Storage Items:

- Database System: PostgreSQL is utilized as the relational database management system (RDBMS) for structured data storage and retrieval.
- Tables: Each major entity corresponds to a PostgreSQL table, ensuring data integrity and efficient querying capabilities.
- Indexes: Indexes are implemented on key fields (e.g., IDs) within PostgreSQL for data retrieval.
- Backup and Recovery: Regular backups and a robust recovery plan are implemented within PostgreSQL to safeguard data integrity and ensure continuity.

This revised section provides a detailed overview of how PostgreSQL is used to manage and organize the major data entities within your Citizen Portal Platform system.

5. Component Design

Each part of the Citizen Portal Platform's architecture has a specific function within the system architecture, which helps the platform run correctly. This is a detailed explanation of what are the functions and their pseudocode briefly:

Function SubmitProposal()

Description: This function allows citizens to submit new proposals for consideration by the city council. The function collects necessary information about the proposal and ensures all required fields are filled out before saving the proposal.

- Inputs: title, description, category, supportingDocuments(String List proposal)
- Outputs: success (the message)

Here is the pseudocode of the function:

```
function SubmitProposal(title, description, category, supportingDocuments):
    if title is empty or description is empty:
        return error "Title and description are required."
 saveProposal(title, description, category, supportingDocuments)
        return success "Proposal submitted successfully."
```

Function: ProvideFeedback()

Description: On this function, citizens can offer their opinions on ideas that have already been made through the Citizen Portal Platform. Before saving the input, it gathers feedback comments and any supporting documentation, making sure all necessary fields are completed.

- Inputs: feedback (includes feedback comments and supporting documents)
- Outputs: success (a message indicating successful submission)

Pseudocode for the ProvideFeedback() function:

```
ProvideFeedback(feedback):
 if feedback is empty:
   return error "Feedback comments are required."
saveFeedback(feedback)
```

return success "Feedback submitted successfully."

Function: CastVote()

Description: Before allowing users access to the platform, this function authenticates them. It compares the supplied credentials—password and username—with user records that are kept in the database. It creates a session token after successful authentication so that the user's session is maintained during their platform interaction.

- Inputs: vote (the citizen’s vote option)
- Outputs: success

Here's the corresponding pseudocode:

```
function CastVote(vote):
    if vote is not in ["approve", "reject", "abstain"]:
        return error "Invalid vote option."
        saveVote(vote)
        return success "Vote recorded."
```

Function: AuthenticateUser()

Description: This function authenticates users before granting them access to the platform. It verifies the provided credentials (username and password) against stored user records in the database. Upon successful authentication, it generates a session token to maintain the user's session throughout their interaction with the platform.

- Inputs: username (the user's username), password (the user's password)
- Outputs: sessionToken (a token to maintain the user's session)

Local Data: userRecord: Contains user details such as username, password hash, and userID. And here's the corresponding pseudocode:

```
function AuthenticateUser(username, password):
 userRecord = getUserRecord(username)
if userRecord is None: return error "User not found."
if not validatePassword(password, userRecord.passwordHash):
 return error "Invalid password."
user sessionToken = generateSessionToken(userRecord.userID)
 return sessionToken
```

Function: TrackProposalStatus()

Description: Within the Citizen Portal Platform, this feature monitors the status of ideas that have been submitted. It obtains a proposal's present state.

- Inputs: none
- Outputs: status (the current status of the proposal)
- Local Data: None

Here's the corresponding pseudocode:

```
function TrackProposalStatus():
   status = getProposalStatus()
   return status
```

Function approveProposal(proposal)

Description: This function is responsible for approving a proposal submitted by a citizen. It takes the proposal object as input and updates its status to "approved" in the database.

- Inputs: proposal (the proposal object to be approved)
- Outputs: status
- Local Data: None

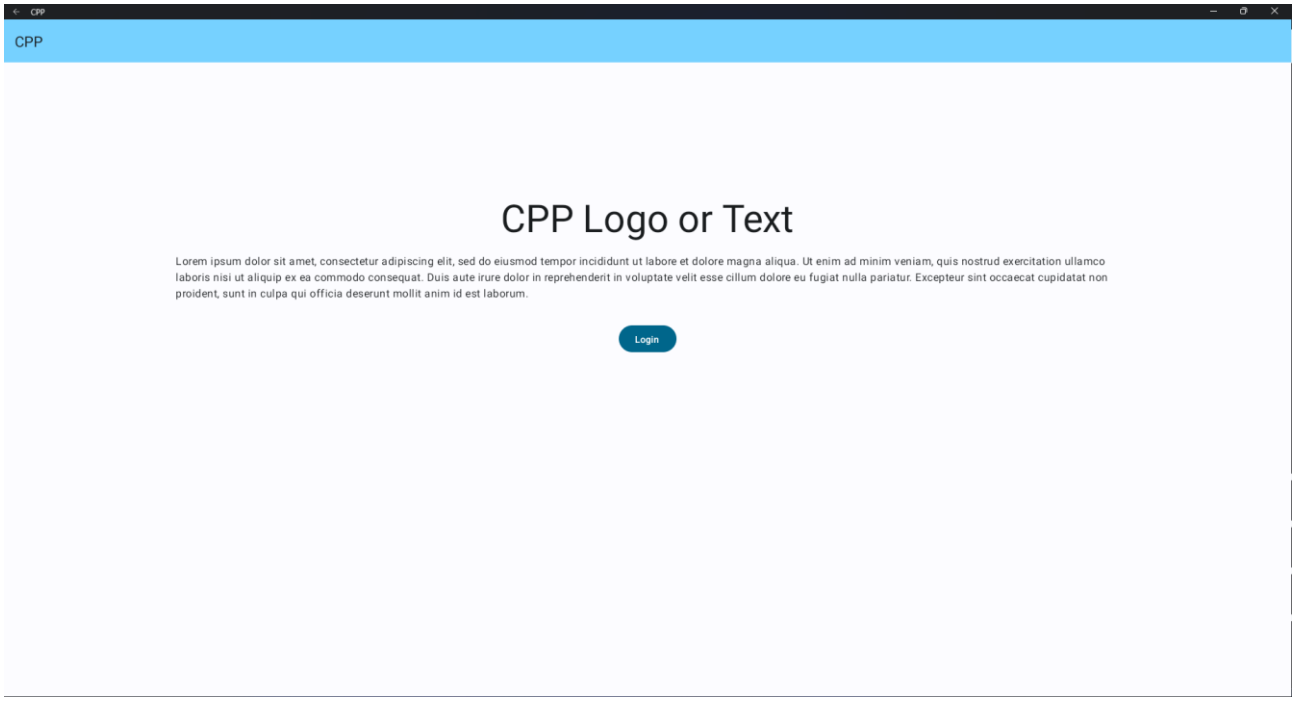
```
function approveProposal(proposal):
    proposal.status = "approved"
    return status
```

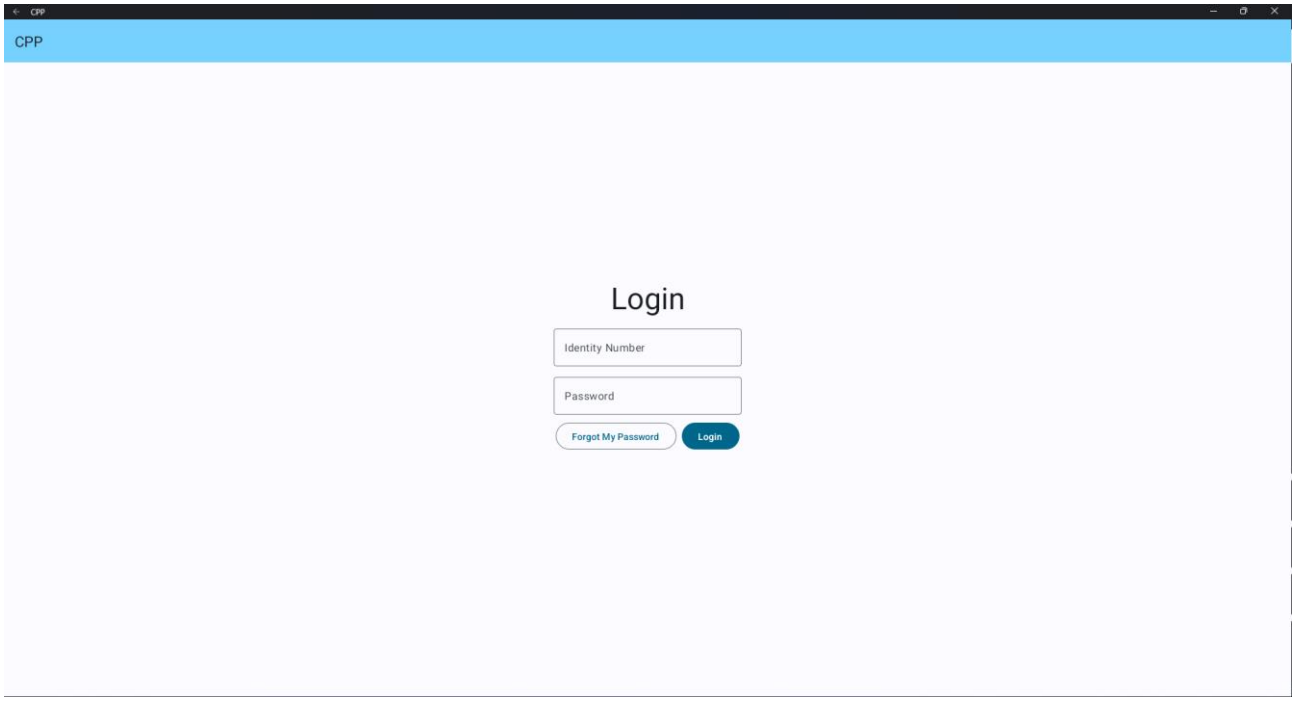
6. Human User Interface Design

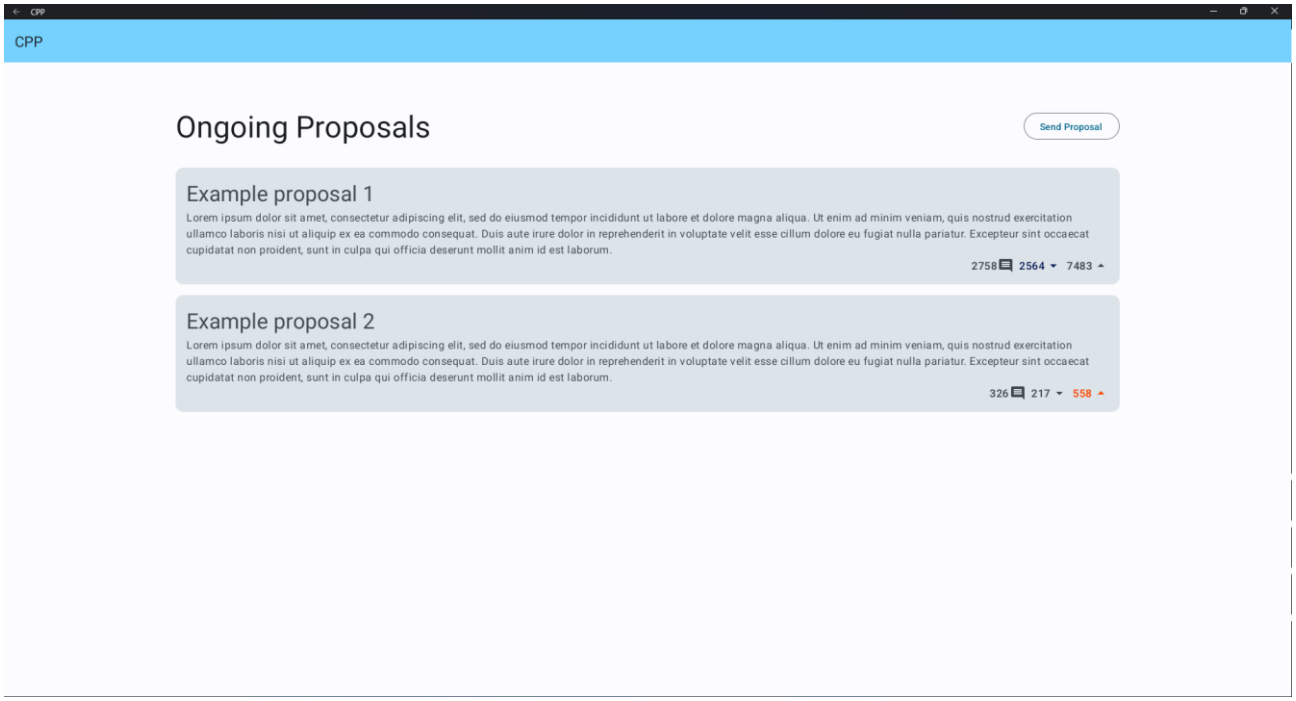
6.1 Overview of User Interface

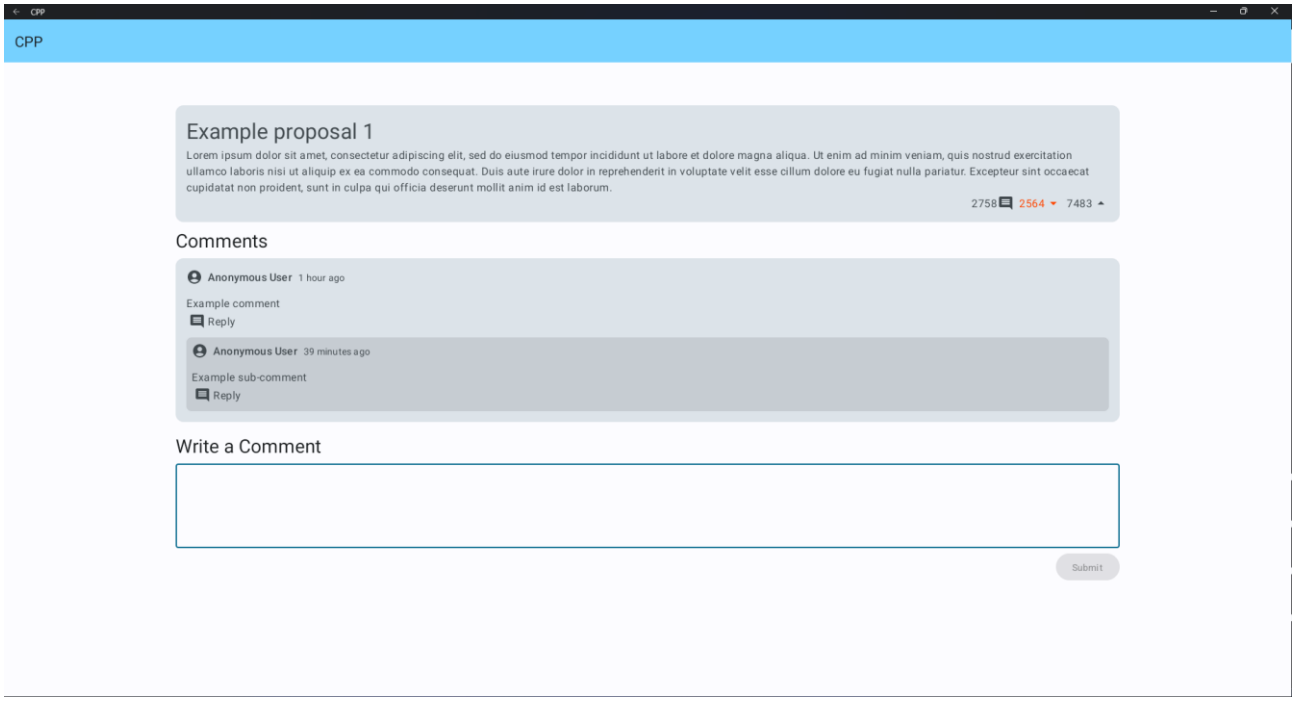
- **Welcome Screen:** Users will be greeted with a welcome message after entering the portal. They can login to portal from this screen.
- **Login Screen:** Users can login to system by entering their identity number and password. After logging in, they are redirected to proposals screen.
- **Proposals Screen:** Users can browse through a list of proposals submitted by the city council and other citizens. They can view details of each proposal, cast votes, see the votes for each proposal, and create their own proposals.
- **Proposal Detail Screen:** Upon selecting a specific proposal, users are presented with detailed information about the proposal. They can view proposal comments, write their own comments, or reply to existing comments.
- **Send Proposal Screen:** Users can submit new proposals by filling out a form with relevant details such as the proposal title, and description. Once submitted, the proposal will be reviewed by the city council for consideration.

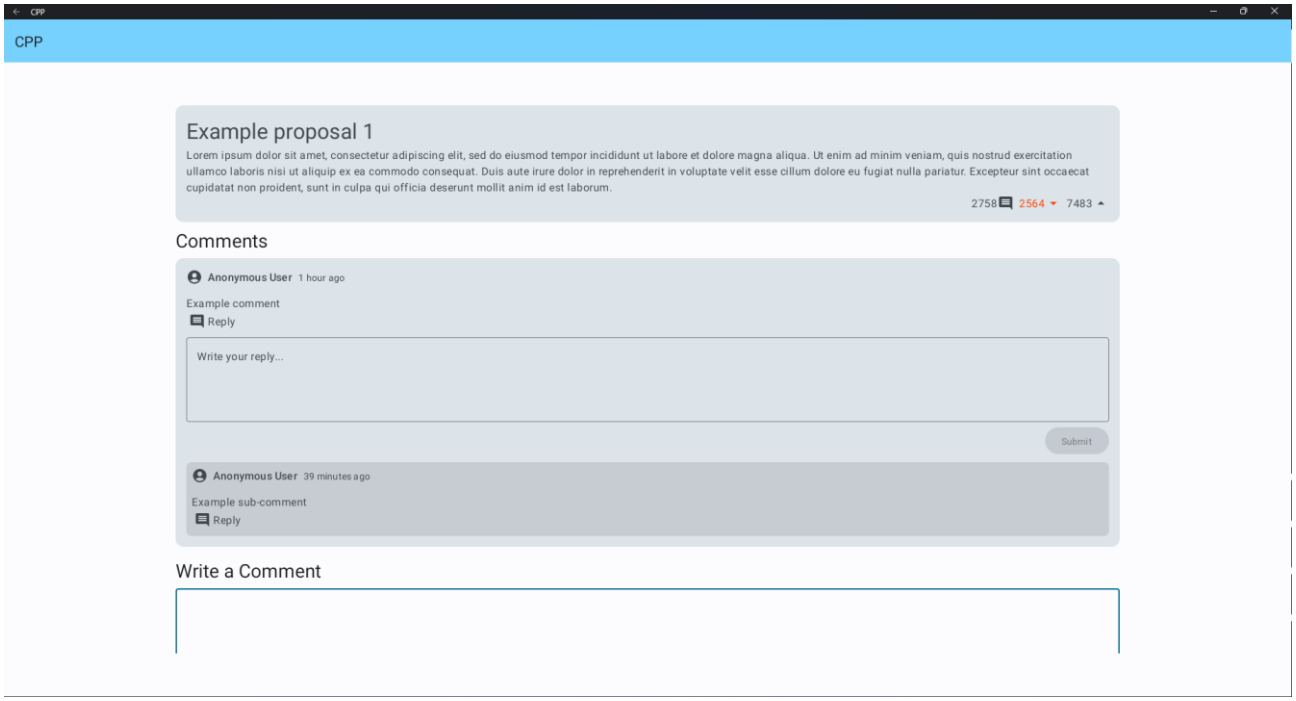
6.2 Screen Images

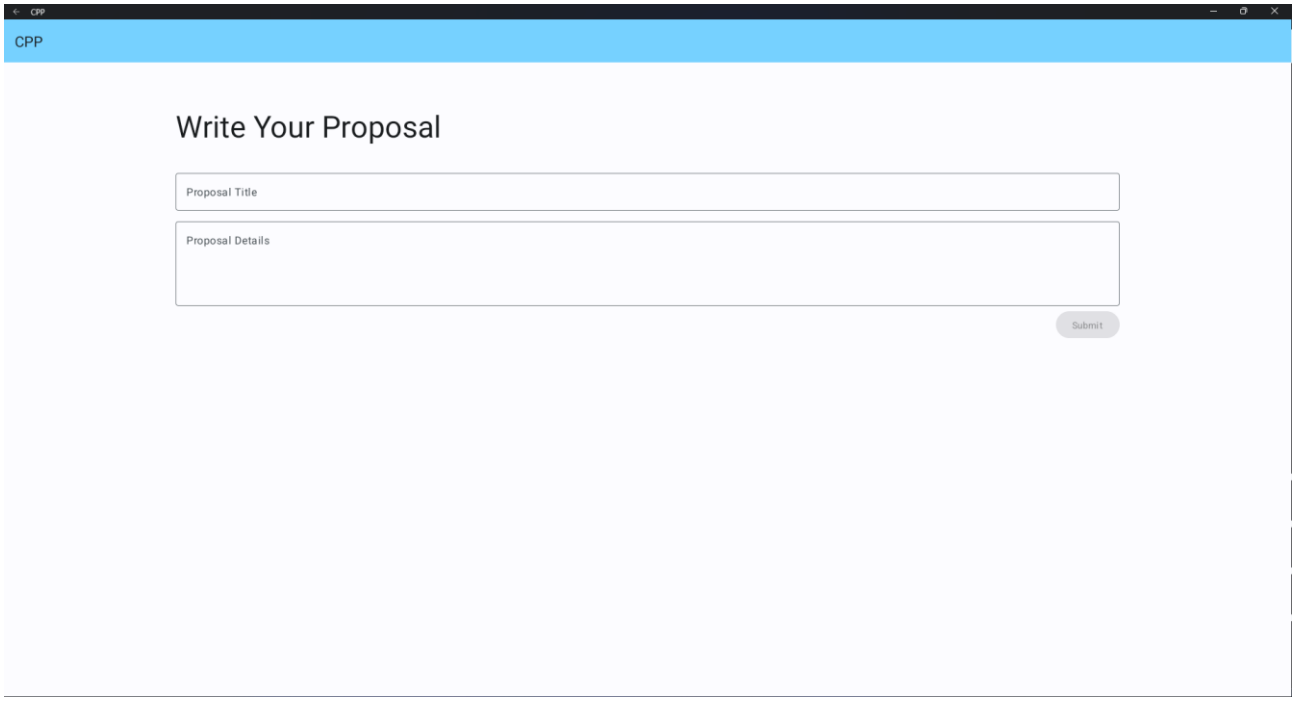

CPP
CPP
CPP Logo or Text
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Login


CPP
CPP
Login
Identity Number
Password
Forgot My Password
Login


CPP
CPP
Ongoing Proposals
Send Proposal
Example proposal 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
2758 2564 7483
Example proposal 2
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
326 217 558


CPP
CPP
Example proposal 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
2758 2564 7483
Comments
Anonymous User 1 hour ago
Example comment
Reply
Anonymous User 39 minutes ago
Example sub-comment
Reply
Write a Comment
Submit





6.3 Screen Objects and Actions

- **Welcome Screen:** Display a welcoming message and include buttons for users to proceed to the login screen.

- **Login Screen:** Provide input fields for users to enter their login credentials. Include buttons for submitting the login form and options for password recovery if they forgot their passwords**.**

- **Proposals Screen:** Display a list of proposals with brief summaries or titles. Include buttons for users to view details of each proposal, cast their votes and submit their own proposals.

- **Proposal Screen:** Display detailed information about the selected proposal, including its title, description, and comments from other users. Include options for users to add their comments and reply to other comments.

- **Send Proposal Screen:** Present a form for users to submit new proposals, including fields for the proposal title and description. Include a button for submitting the proposal.

7. Requirements Matrix

7.1 Components

Component	Description
User Interface (UI)	The system provides citizens with a platform to view ongoing proposals and select individual proposals for feedback and voting.
Feedback Submission	Allows users to submit feedback comments for each proposal, with an option to attach supporting documents.
Feedback Validation	Validate feedback submissions to ensure all required fields are filled out and data is in the correct format.
Submission Confirmation	Notifies the user upon successful submission of feedback, confirming receipt.
Voting Options	Provides users with options to express their preference on each proposal, including approve, reject, or abstain.
Vote Recording	Records and tallies vote for each proposal, updating the proposal status accordingly.
Error Handling	Manages errors during the feedback or voting, displaying appropriate error messages to the user.

7.2 Data Structures

Data Structures	Description
Proposal	Contains information about each proposal, including its title, description, status, and associated feedback and votes.
Feedback	Stores feedback comments submitted by users, including any attached supporting documents.
Vote	Represents individual votes cast by users, indicating their preference (approve, reject, or abstain) for a proposal.

7.3 Cross Reference

The following table cross-references the system components and data structures to the functional requirements specified in the SRS document:

Functional Requirements	**System Component / Data Structure**
REQ-1	User Interface (UI), Proposal
REQ-2	Feedback Submission, Feedback
REQ-3	Feedback Validation
REQ-4	Submission Confirmation
REQ-5	Voting Options, Vote
REQ-6	Vote, Proposal
REQ-7	Error Handling