Table of Contents

CHAPTER 1: INTRODUCTION

1.1 Background

Citizens haven’t had a suitable space to voice their opinions. Usually done on social media platforms, which isn't very effective, but we have a system called “Naagarik Feedback” for citizen’s feedback.

The “Naagarik Feedback” (citizen feedback system) project aims to provide a platform for citizens to offer feedback on their localities. This initiative empowers residents to voice their opinions, report issues, and share positive experiences about their living areas. Feedback includes the citizen's full name, contact number, and the area they reside in, along with columns for positive and negative feedback. This system fosters community engagement and allows local authorities to address concerns and improve services.

1.2 Problem Definition

In an era where digitization is transforming every aspect of our lives, the Naagarik Feedback system is created to allow citizens to provide their valuable opinions about their localities digitally. However, implementing such a system comes with several concerns that need to be addressed properly.

One of the main challenges is filtering out fake reviews submitted by genuine users. The system administrators and relevant authorities who handle and resolve these issues may face serious challenges because of these misleading inputs. To keep the system's integrity intact, feedback must be guaranteed to be real. The effectiveness of the system may be seriously challenged in the absence of strong mechanisms for discriminating between legitimate feedback and spam or fraudulent inputs, which could result in inefficiencies.

In addition, users must be assured that the created system is reliable and will operate as intended. This is because many individuals lack confidence in the online system that local government agencies and other relevant authorities rely on. Residents in areas where the system is being used must be sure that their opinions and feedback are being observed and evaluated properly by the responsible authorities.

Finally, transforming collected feedback into actionable insights for local authorities is another critical challenge. The system must be capable of analyzing feedback efficiently and providing clear, actionable recommendations. This requires sophisticated data processing and analysis tools that can handle large volumes of feedback and distill them into meaningful insights that can guide decision-making and policy formulation.

Overall, while the Naagarik Feedback system offers significant benefits, it is crucial to ensure that the platform is reliable and functions properly. By addressing these concerns, we can help ensure that citizens' opinions and feedback are effectively observed and evaluated by the responsible authorities, thereby fostering greater trust and participation in local governance.

1.3 Objective

The objectives of this project are:

- To provide a platform for residents to voice their concerns and suggestions.
- To Enable local authorities to respond to feedback and improve public services.
- To Implement methods to ensure the authenticity and relevance of feedback.

By achieving these objectives, the project aims to create a more responsive and community-driven approach to local governance.

1.4 Scope

The scope of this project includes the implementation and evaluation of feedback standards and user experience improvements in the Naagarik Feedback system.

This project will focus on addressing the concerns associated with the Naagarik Feedback system, such as verifying user authenticity, ensuring data privacy, and transforming feedback into actionable insights. The project will also aim to enhance the overall user experience of the platform by improving the ease of use, reliability of feedback information, and user convenience

To achieve these goals, the project will involve the following activities:

- Review existing feedback systems: Analyze current feedback systems to identify strengths and weaknesses.
- Ensure proper feedback evaluation: Implement processes to thoroughly check and validate user feedback.
- Provide transparency: Ensure users receive proper updates regarding the status and impact of their feedback.
- Develop user experience improvements: Enhance the platform's usability and accessibility.
- Evaluate feedback processing: Assess the effectiveness of feedback collection and analysis methods.

1.5 Limitations

The Naagarik Feedback system has several drawbacks that may affect both user satisfaction and overall efficiency, despite its apparent advantages. These constraints are caused by several things, such as performance limitations, technical complexities, and the requirement for customization to meet a range of user requirements.

- KYC System Not Implemented: The system does not include a Know Your Customer (KYC) process due to its complexity, making it difficult to verify if users are genuinely residing in the specified area.

- Performance Issues: When there is a significant volume of traffic, the application may experience performance problems.
- Lack of Customization: Pre-built queue management systems may not offer the customization needed for some businesses.

Here are some additional limitations of the Naagarik Feedback system:

- **Privacy Concerns:** There may be privacy issues with the gathering and use of personal data, such as names and contact information. It's critical to maintain a balance between the necessity for feedback verification and data security.
- **User Behavior:** The effectiveness of the feedback system depends on user behavior. Despite the implementation of feedback standards and protocols, some users may provide false or misleading feedback, compromising the system's integrity.
- **Cost Limitations:** A substantial amount of money could be necessary for the implementation of this feedback system, which could cause difficulties for smaller local bodies or concerned authorities.
- **Geographic Limitations:** Implementation of feedback system may be limited by geographic factors, such as infrastructure availability or varying local issues.
- **Regulatory Restrictions:** Legal and regulatory restrictions may apply to the system, which may restrict the use of specific feedback guidelines and user experience enhancements.
- **Authority Cooperation:** Local authorities may need to cooperate to implement feedback standards and user experience improvements, as they may have conflicting goals or be opposed to change.

1.6 FEASIBILITY STUDY

It is beneficial and essential to evaluate the feasibility of a project as soon as feasible. The Naagarik Feedback system's viability was evaluated through the following feasibility studies:

1. **Operational Feasibility:** Regardless of /their level of technical expertise, all residents must be able to easily navigate and operate the system. The suggested system has an easy-to-use interface, clear instructions, and a straightforward architecture. It should make it simple for feedback to be submitted and enable immediate responses from local government. The system's operational reach must guarantee that people with no prior computer experience may effectively explore and utilize it.
2. **Technical Feasibility:** This involves analyzing the system's limitations and technological requirements. It is important that the suggested system be constructed with flexible, expandable technologies capable of managing the expected amount of feedback. It should provide capabilities like data encryption, user identification feedback processing and interact seamlessly with the current local authority system. The availability of qualified employees to create, maintain, and upgrade the system is another factor in technical viability.
3. **Economic Feasibility:**

   It's used to evaluate the cost-effectiveness and financial viability of the project. This critical evaluation includes a thorough examination of numerous financial aspects to guarantee that the project is both financially viable and able to provide value. It consists of start-up expenses, development expenditures, running costs, income creation, and ROI calculations.

4. **Schedule Feasibility:**

   This part evaluates the project's ability to be finished in the allotted time. It involves evaluating the project plan to make sure that all work can be completed on time, taking into account critical due dates and objectives. Potential delays can be found and eliminated by carefully preparing and keeping an eye on the project schedule, which guarantees timely project completion.

A Gantt chart has been created in order to efficiently manage and monitor the advancement of the Naagarik Feedback system project. The Gantt chart shows the project schedule, with important duties and objectives highlighted along with the start

and finish dates of each phase. This graphic depiction of the project schedule guarantees that all tasks are in line with the overall plan and keeps stakeholders updated on the project's status.

1.7 Gantt Chart

A Gantt chart outlines the project timeline and tasks:

	Task Name	Duration	Start	Finish	Predec
1	**Pase 1-Requirement Analysis**	**24 days**	**Tue 5/13/25**	**Fri 6/13/25**	
2	Identiftiying Project's Problems	5 days	Tue 5/13/25	Mon 5/19/25	
3	Evaluating Feasibility	7 days	Tue 5/20/25	Wed 5/28/25	2
4	Develop Project Plan	6 days	Thu 5/29/25	Thu 6/5/25	3
5	System Requirement	6 days	Fri 6/6/25	Fri 6/13/25	4
6	**Phase 2- Design**	**14 days**	**Mon 6/16/25**	**Thu 7/3/25**	**1,5**
7	Logical Design	3 days	Mon 6/16/25	Wed 6/18/25	
8	Physical Design	7 days	Thu 6/19/25	Fri 6/27/25	7
9	Database Design	4 days	Mon 6/30/25	Thu 7/3/25	8
10	**Phase 3- Testing**	**5 days**	**Fri 7/4/25**	**Thu 7/10/25**	**9**
11	Junit Testing	3 days	Fri 7/4/25	Tue 7/8/25	6
12	Selenium Testing	2 days	Wed 7/9/25	Thu 7/10/25	11
13	**Phase 4- Implementation and Maintainance**	**20 days**	**Fri 7/11/25**	**Thu 8/7/25**	**12,10**
14	Testing and Bug Fixing	7 days	Fri 7/11/25	Mon 7/21/25	
15	Deployment and Launch	7 days	Tue 7/22/25	Wed 7/30/25	14
16	Post- Launch Maintainance	6 days	Thu 7/31/25	Thu 8/7/25	15

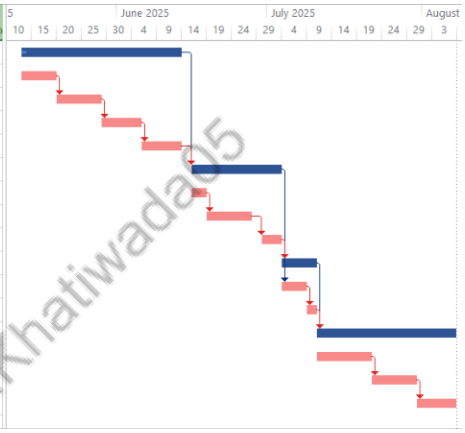


Figure 1.1: Gantt Chart of Naagarik Feedback System

Explanation:

Planning Phase

- Identifying and Defining the Project's Problem: This step involves determining the main problem or requirement that the project is trying to solve. A clear problem definition guarantees that all parties involved are aware of the project's goals and parameters and sets the context for all following project activities.
- Evaluating Feasibility: In order to determine whether the project is feasible from a technical, financial, and operational position, feasibility studies must be carried out. It assists in recognizing the limitations and resources needed to fulfill the project's objectives.
- Developing Project Plan, ROI, and NPV Calculation: Developing a thorough project plan that includes resources, deadlines, and tasks. Additionally, comparing the project's expenditures and advantages financially is made easier by computing the Net Present Value (NPV) and Return on Investment (ROI).

Analysis Phase

- System Requirements: Obtaining specific needs from stakeholders is the first stage in determining what the system should be able to accomplish. This covers both non-functional (how the system should operate) and functional requirements (what the system should be able to achieve).
- Description of Module: breaking the system into simpler, smaller parts or components. Every module is thoroughly explained, including information on its unique features and how it works with other modules.
- System Flow Diagram: Generating diagrams that show how information moves through the system and how operations are carried out. These diagrams aid in illustrating the intended operation of the system and pointing out any possible problems with the process.

Design Phase

- Logical Design: Creating models that reflect the system's functioning without concentrating on the specifics of its physical implementation, such as entity-relationship diagrams and data flow diagrams, is the process of developing the system's logical structure.
- Physical Design: embedding physical specifications into the logical design. Determining the hardware, software, network infrastructure, and other physical components required for system implementation falls under this category.
- Database Design: Creating the database schema that will house the data for the system. To ensure efficient access and data integrity, this involves creating tables, fields, relationships, and limitations.
- Normalization and Transformation of E-R Diagram into Relation: Making sure the database architecture is standardized to remove unnecessary components and enhance data accuracy. converting relational database models that may be used in a database management system from entity-relationship diagrams.

Implementation and Maintenance Phase

- Testing and Bug Fixing: identifying and resolving any bugs or issues in the system by carrying out a variety of tests, including unit, integration, system, and user acceptability testing. confirming that the system satisfies the requirements and performs as intended.
- Deployment and Launch: putting the system online and ready for user use in production. To guarantee a successful launch, this involves setting up the production environment, transferring data, and carrying out last-minute inspections.
- Post-Launch Maintenance: After the system becomes operate, continuing support and maintenance are provided to handle any problems that may occur. This include applying updates, resolving bugs, and refining the product in light of customer input.

CHAPTER 2: SYSTEM ANALYSIS

The process of gathering and analyzing data, recognizing issues, and breaking down a system into its constituent parts is known as system analysis. This approach is used to analyze a system or its components to determine its goals. It is a method of solving problems that makes the system better and guarantees that every part functions effectively to fulfill its intended role.

2.1 Functional Requirement for Naagarik Feedback:

Functional requirements outline a system's behavior and operations, including what it should do and how it should react to inputs. Users must be able to send feedback using an online form for the Naagarik Feedback System to function.

These requirements are essential for defining the functionalities and features that the system must possess to meet user needs and achieve its objectives.

1. User Registration:
    - It should be possible for users to register using their name, phone number, email address, and other required information.
    - When a user registers, the system needs to authenticate and verify their details.
2. Feedback Submission:
    - It should be possible for users to provide feedback, including choosing the feedback type (complaint, recommendation, query, etc.) and supplying specifics.
    - Validation should be included in feedback submissions to guarantee accuracy and completeness.
3. Feedback Management:
    - Admins should be able to examine, classify, and assign feedback to the relevant departments for resolution.

- Input should be able to be arranged by admins based on its importance and necessity.

4. Response Management:
    - Responses to user input should be drafted and sent by administrators to users.
    - For reference, responses must be recorded and linked to the initial feedback.
5. User Interface:
    - The system should have a user-friendly interface that makes it simple to navigate and submit feedback.
    - User-friendly tools for effectively managing comments and responses should be included in admin interfaces.

2.2 Non-Functional Requirement Naagarik Feedback:

Non-functional requirements outline a system's quality attributes that specify how it should function, including security, dependability, performance, and usability. The Naagarik Feedback System's ability to manage several concurrent user contributions without experiencing performance deterioration is a non-functional requirement.

These requirements are critical for ensuring that the system not only functions correctly but also provide a satisfactory user experience and meets performance standards.

1. Performance:
    - Many concurrent users should be supported by the system without noticeably affecting performance.
    - User activity response times, such as registration and feedback submission, should fall within acceptable limits.
2. Security:

- To avoid unwanted access, user data (personal information, feedback details, etc.) should be encrypted and stored securely.
- Strict restrictions on admin rights and access to private information should be implemented via permission and authentication systems.

3. Reliability:
    - There should be minimal downtime and the system should be up and running 24/7.
    - Data integrity needs to be maintained to avoid responses and feedback from being lost or misrepresented.
4. Scalability:
    - The system needs to be scalable to handle future increases in the number of users and comments.
    - It should facilitate an effortless expansion of database and server capacity as required.
5. Usability:
    - Users with different levels of technical expertise should be able to utilize and understand the user interface.
    - Successful process management and decision-making procedures should be facilitated by administrative tools.
6. Compliance:
    - The system is bound by industry standards and applicable data protection laws.
    - For safety considerations, it must allow transparency and user action monitoring.

2.3 Context Diagram:

A context diagram is a high-level, graphical depiction of a system that demonstrates how it interacts with external elements and is shown as a single process. It offers a brief overview of the boundaries of the system, the entities it interacts with, and the main

data flows between the system and these outside entities. As a first stage in system analysis, this graphic aids in understanding the system's scope and surroundings.

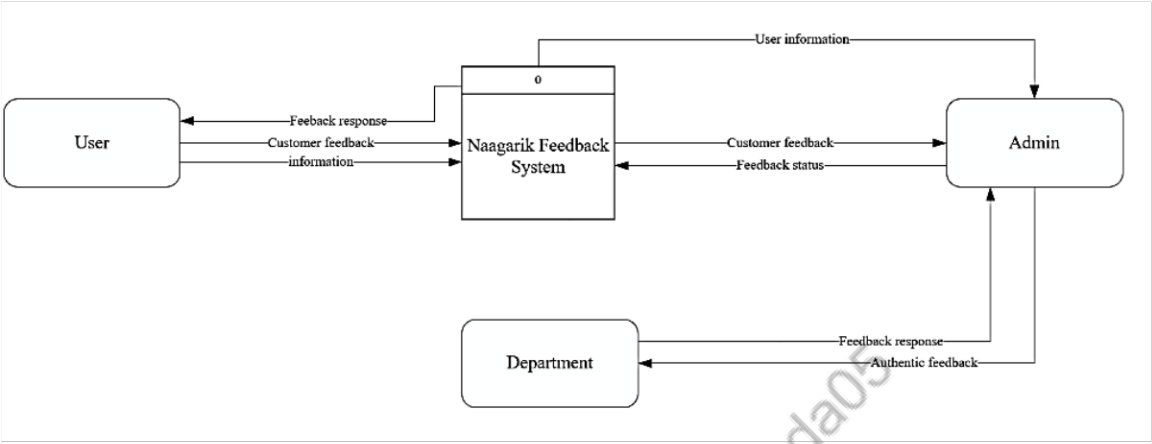


Figure 2.1: Context diagram

2.4 Level 0 DFD:

A Level 0 Data Flow Diagram (DFD) illustrates the core functions and their interactions inside the system by breaking the main process of the system into significant sub-processes. In comparison to the context diagram, it offers a more thorough perspective by highlighting important activities, data stores, and data flows. Understanding the fundamental functions of the system and their connections is made easier by this diagram.

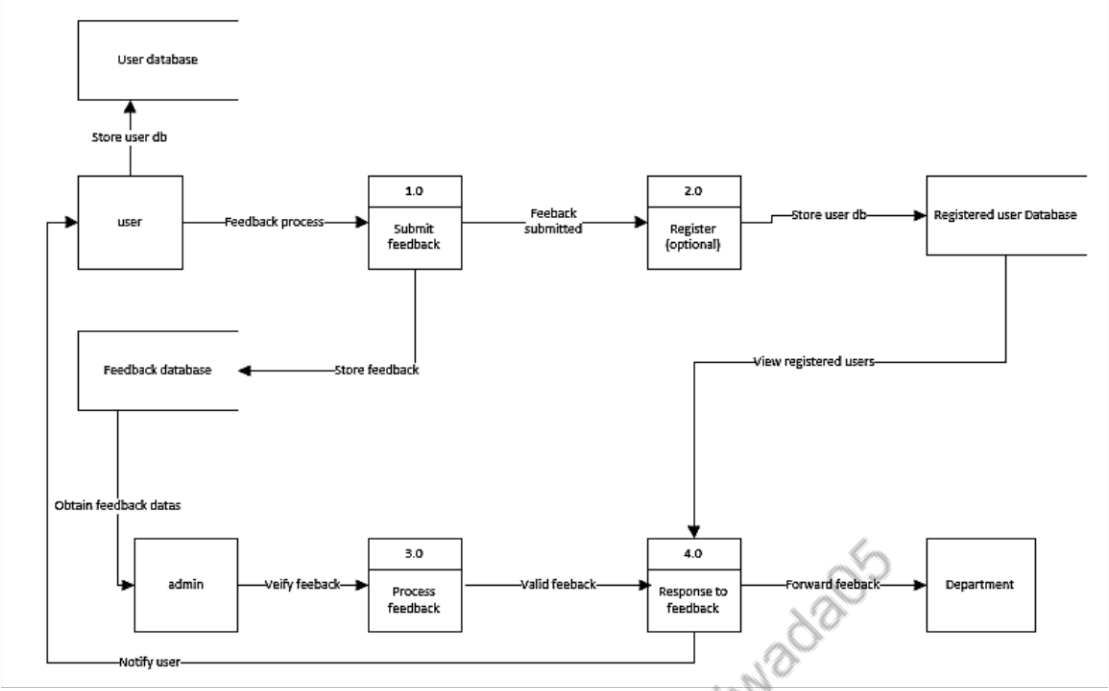


Figure 2.2: Level 0 DFD

2. 5 Entity Relationship Diagram:

An example of a data model is an entity-relationship (ER) diagram, which shows the relationships between the entities (items of interest) in a system or organization. It's frequently used to show the logical structure of databases in database design.

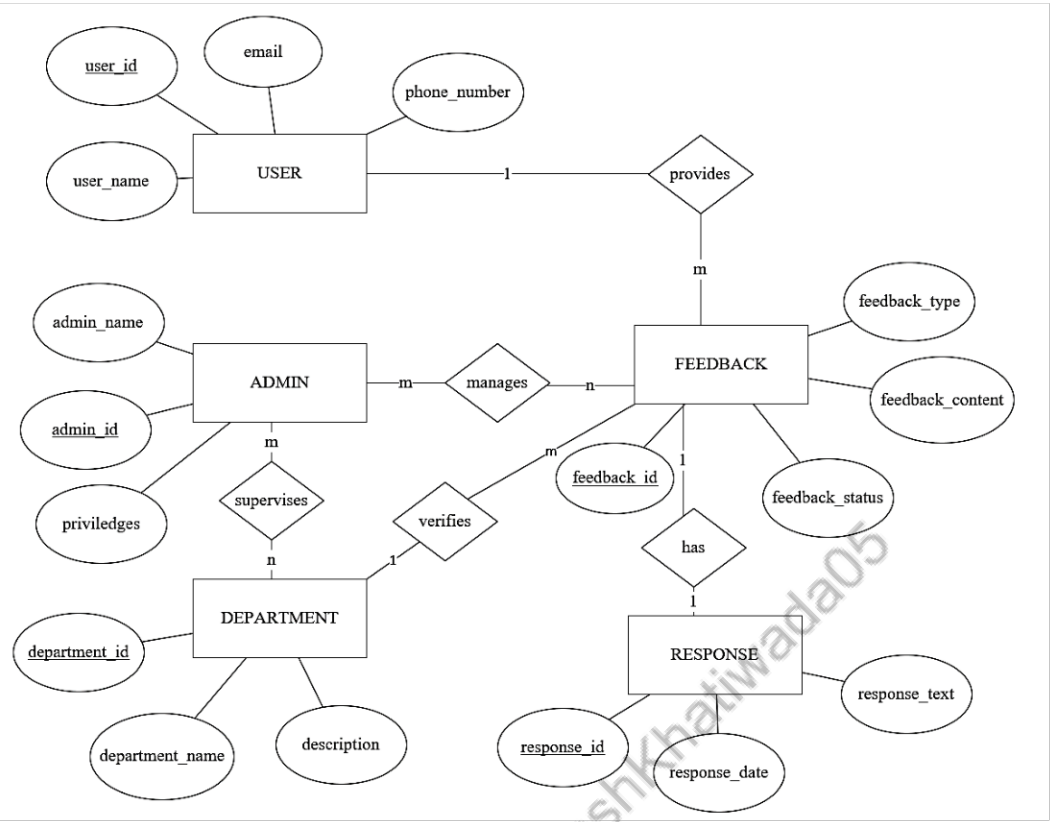


Figure 2.3: Entity Relationship Diagram of Naagarik Feedback

CHAPTER 3: DESIGN

Software design is a technique that aids in the creation and implementation of software by transforming user requirements into a form that is appropriate. It deals with expressing the requirements of the client in a format that can be readily

implemented using computer language, as stated in the SRS (Software Requirement Specification) document.

3.1 Relational Table:

A relational database's basic building block is a relational table, which arranges data into rows and columns. In a relational database, every table denotes a type of entity, and every row, also known as a record, in a table denotes an instance of that object. A table's columns, sometimes called fields, stand for the entity's attributes. Primary keys are used in relational tables to uniquely identify each row, whereas foreign keys are used to create associations between tables.

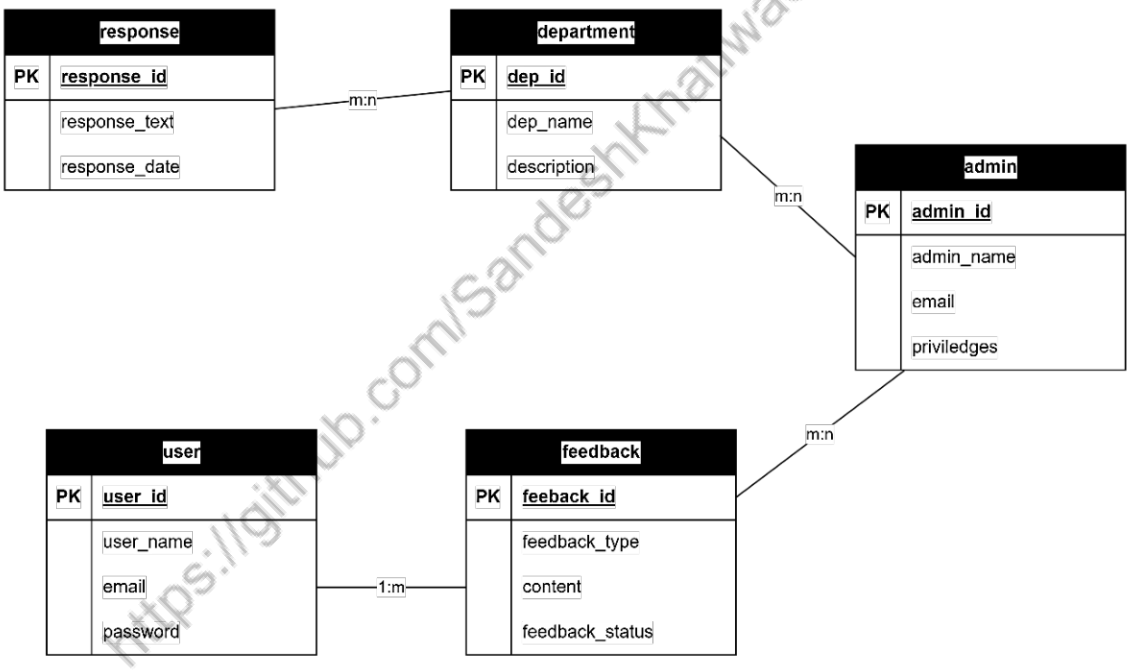


Figure 3.1: Relational table of Naagarik Feedback

3. 3 Physical File and Database design:

The database schema, which includes table definitions, keys, and associations, will be outlined to construct a physical file and database design for the Naagarik Feedback System. SQL statements based on this architecture will be used to create the database

tables.

Database Schema

1. User Table
2. Admin Table
3. Feedback Table
4. Response Table
5. Department Table

Database for Naagarik Feedback System

Relational database management systems would be the most suitable kind of database for the Naagarik Feedback System, considering the needs and the nature of the data (RDBMS).

Reason for choosing RDBMS:

1. Structured Data: The information is very well-organized, with distinct entities (Department, Feedback, Admin, and User) and their relationships.
2. ACID Properties: To guarantee data consistency and integrity, the system probably needs robust transactional support (Atomicity, Consistency, Isolation, Durability).
3. International Key Restraints: the maintenance of referential integrity between tables using foreign key relationships.
4. SQL Support: The necessity of using SQL for data management and querying.
5. Standardization: RDBMSs are extensively utilized and comprehended, offering a solid and established atmosphere for advancement and upkeep.

RDBMS examples that can be used for Naagarik Feedback is Oracle Database.

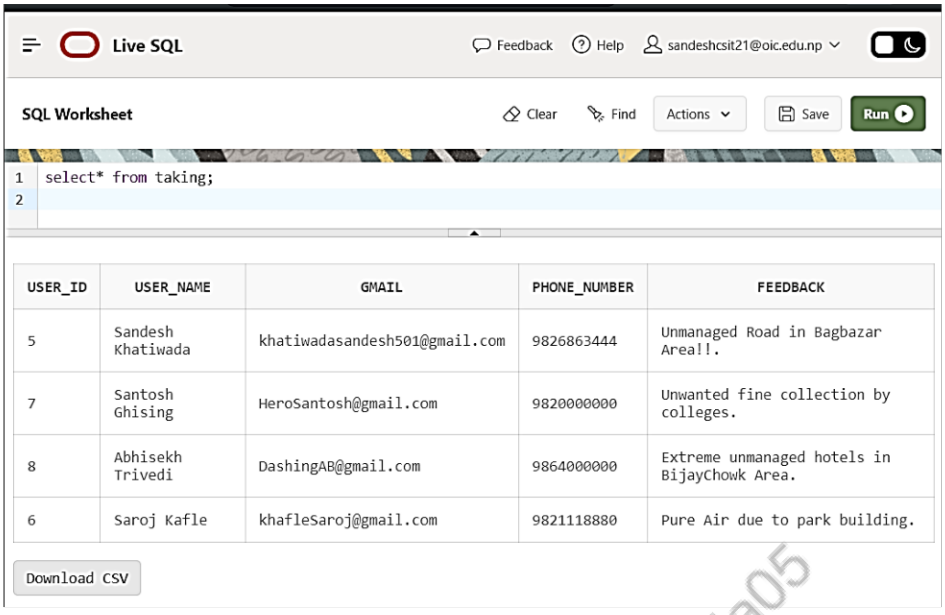


Figure 3.2: Oracle Database used for Naagarik Feedback

3. 4 Form and Interface:

A form in the context of web applications and information systems is an interactive page or interface element that allows users to input, submit, and manage data. Forms are used to collect information from users, which is then processed and stored by the system. They are fundamental components of user interfaces, facilitating interaction between the user and the system.

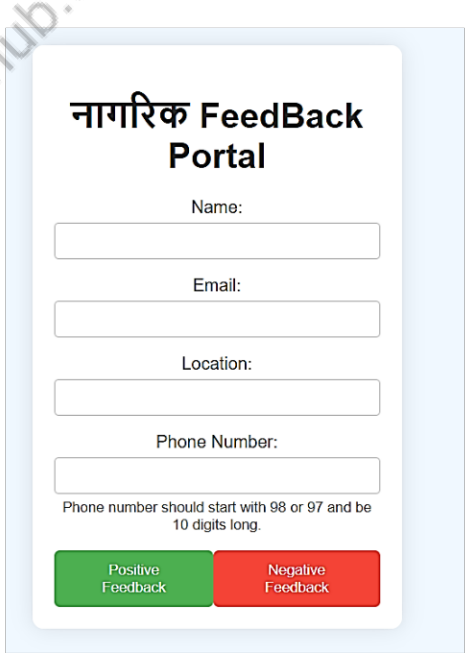
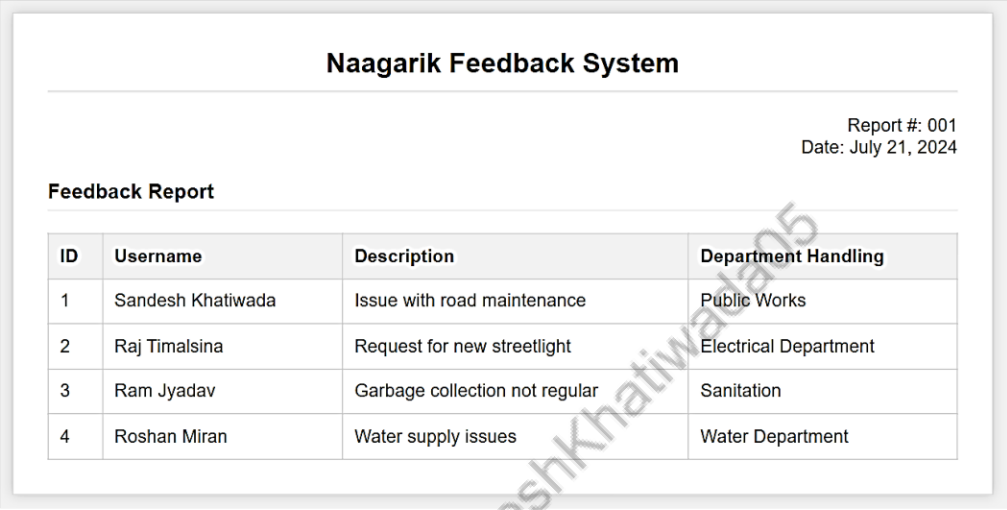


Figure 3.3: Form of Naagarik Feedback

3.5 Report:

A key component of software development, forms are used to gather user input and make it easier for users to engage with the program. In addition to functionality, a well-designed form should also be accessible and user-friendly so that users can enter data quickly and easily.



Naagarik Feedback System

Report #: 001
Date: July 21, 2024

Feedback Report

ID	Username	Description	Department Handling
1	Sandesh Khatiwada	Issue with road maintenance	Public Works
2	Raj Timalsina	Request for new streetlight	Electrical Department
3	Ram Jyadav	Garbage collection not regular	Sanitation
4	Roshan Miran	Water supply issues	Water Department

Figure 3.4: Report for Naagarik Feedback

CHAPTER 4: OBJECT ORIENTED DESIGN

The system is viewed as a collection of objects (also known as entities) in the object-oriented design a model. Each object manages its state data separately, and the state is shared among the objects. To provide modularity and concern separation, objects contain both the data and the functions that operate on it.

For example: Several entities, including User, Admin, Feedback, and Department, are modeled as objects in the Naagarik Feedback System. In order to effectively handle and process input, these items communicate with one another.

4.1 UML Diagram:

The Unified Modeling Language (UML) can be used to visualize software and systems through UML diagrams. To comprehend the designs, code architecture, and suggested implementation of complex software systems, software engineers generate UML diagrams.

It is of Two Types:

a) Structural Diagram
b) Behavior Diagram

4.1.1 Structural Diagram:

Structural UML diagrams show how a system is organized by showing its constituent parts, like classes, objects, and packages. They stand for the constituent parts of the system as well as their interrelationships.

1. Class Diagram
    - Every object-oriented technique starts with a class diagram, which may be used to illustrate classes, relationships, interfaces, associations, and cooperation. Class diagrams use standardized UML.
    - Classes have a suitable structure to represent classes, inheritance, relationships, and everything that OOPs have in their context because classes are the building blocks of an
      OOPs-based program.

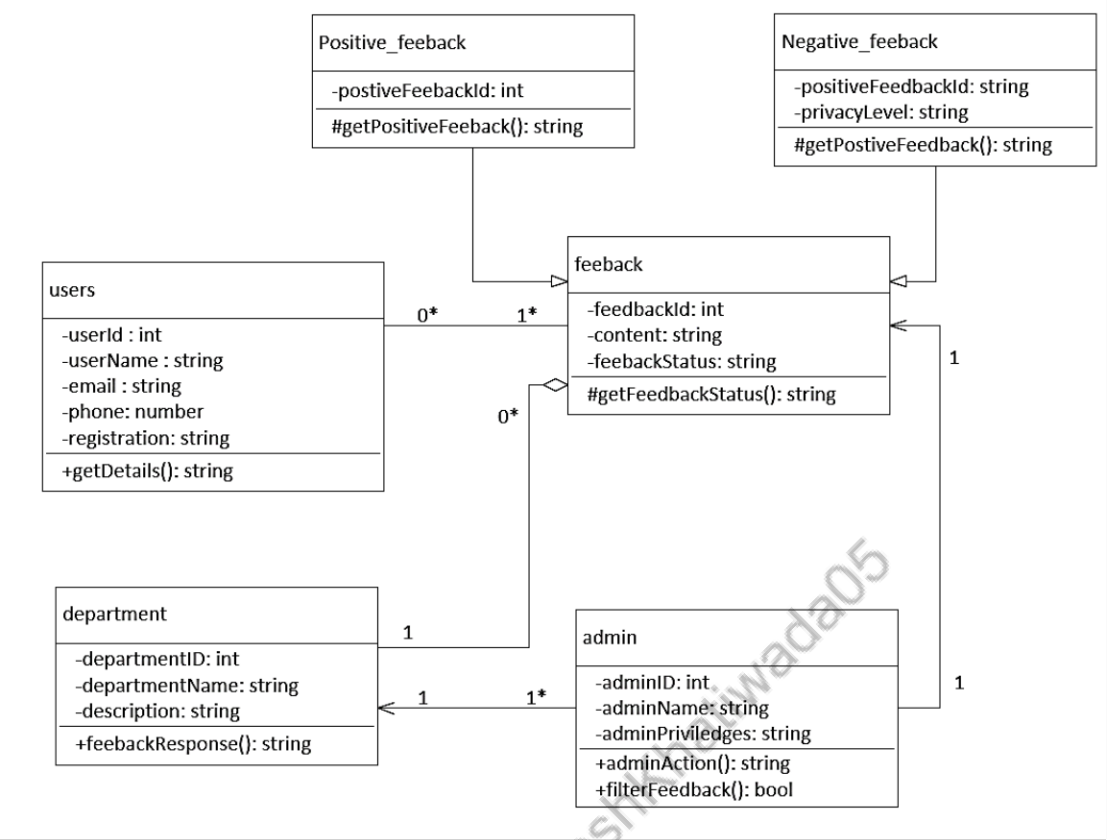


Figure 4.1: Class diagram for Naagarik Feedback

2. Object Diagram

- You may think of an Object Diagram as a screenshot of the instances within a system and their relationships.
- We may examine the system's behavior at a certain moment since object diagrams show behavior when objects have been instantiated.
- To illustrate and comprehend the functional needs of a system, object diagrams are essential. Put differently, an object diagram in the context of the Unified Modeling Language (UML) is a diagram that depicts a full or partial perspective of a system's structure at a certain point in time

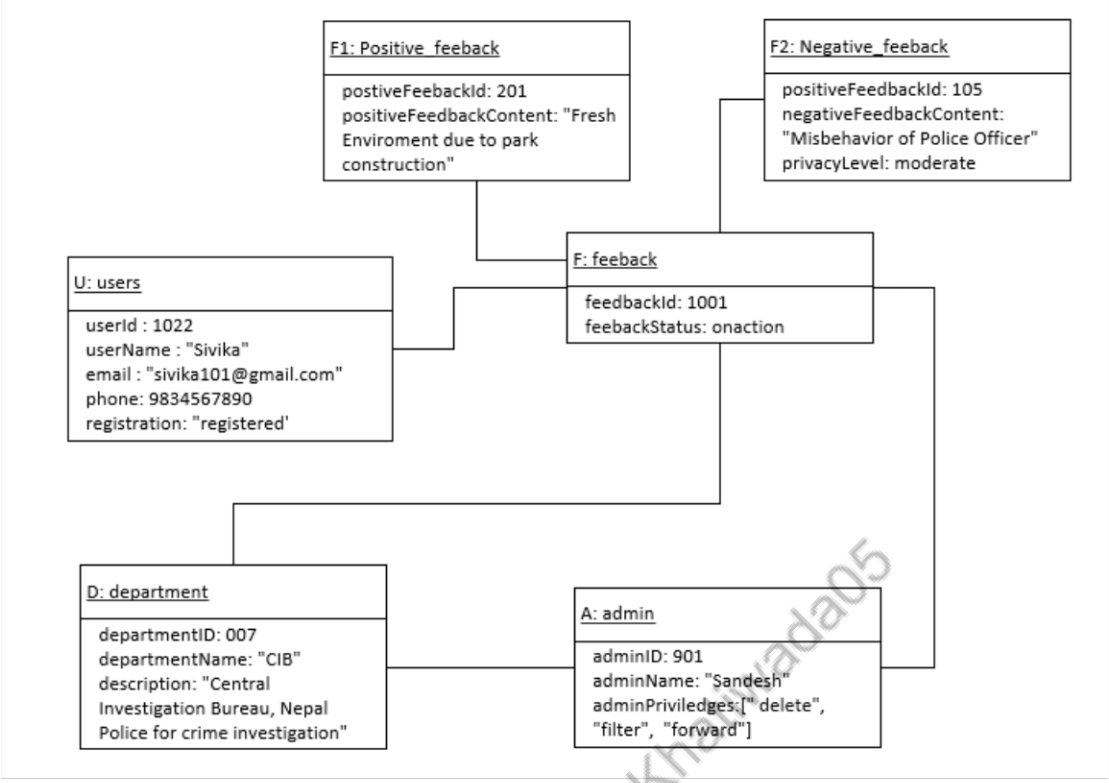


Figure 4.2: Object Diagram for Naagarik Feedback

3. Component Diagram

- The arrangement of a system's physical components is shown using component diagrams. We employ them in the modeling of implementation specifics.
- Component diagrams show the structural relationships between the various components of a software system and enable us to determine whether planned development has addressed all of the functional requirements.
- The usage of component diagrams becomes crucial throughout the design and construction of complicated systems.
- System components communicate with one another using interfaces.

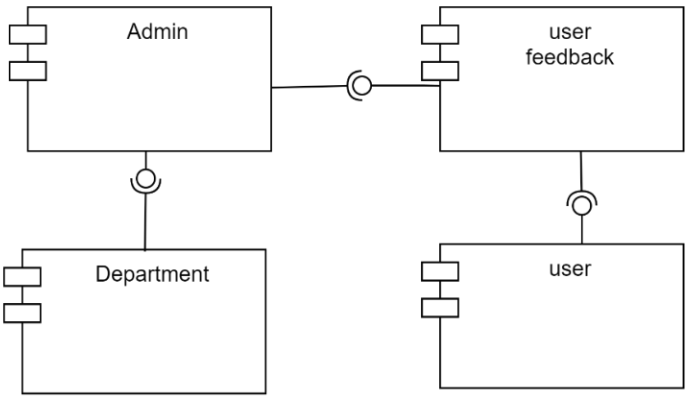


Figure 4.3: Component Diagram for Naagarik Feedback

4. Deployment diagram

- Implementation The hardware and software of a system are represented through diagrams. It provides information about the hardware and software components that are compatible with one other.
- We use the distribution of software artifacts over distant targets as an example of system architecture.
- They are mainly utilized for distributing or deploying software across several workstations with various setups.

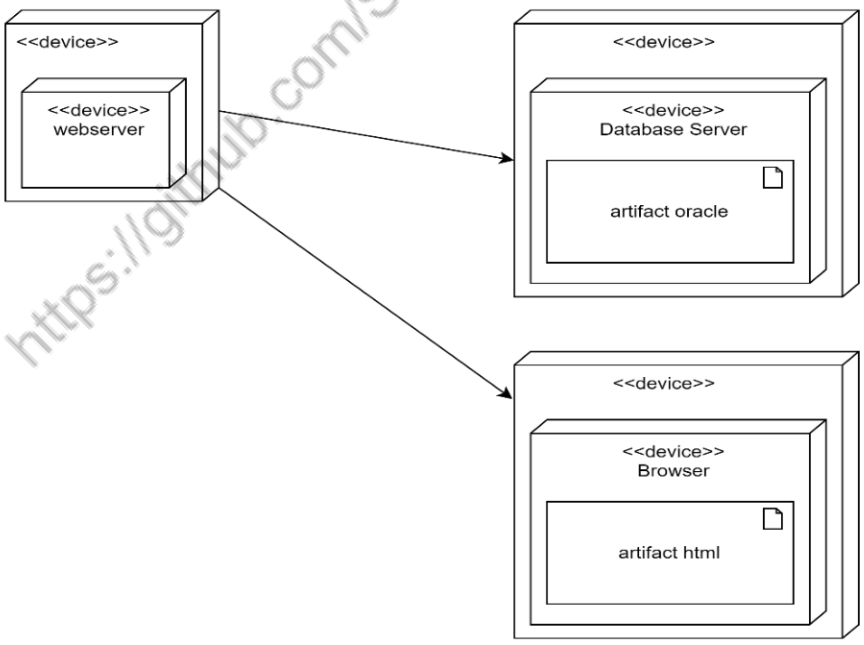


Figure 4.4: Deployment diagram for Naagarik Feedback

4.1.2 Behavioral Diagram:

Behavioral UML diagrams concentrate on displaying a software system's dynamic features, such as its behavior, response to external stimuli, and state changes that occur

during runtime.

Types of Behavioral UML diagrams

1. State Machine Diagrams

A state diagram is used to show how a system or a portion of a system is configured at specific points in time. Finite state transitions are used in this behavioral diagram to depict the behavior. State-chart diagrams and state machines are other names for state diagrams. These phrases are frequently used synonymously. In other words, a state diagram is used to simulate how a class would behave dynamically over time in response to shifting external stimuli.

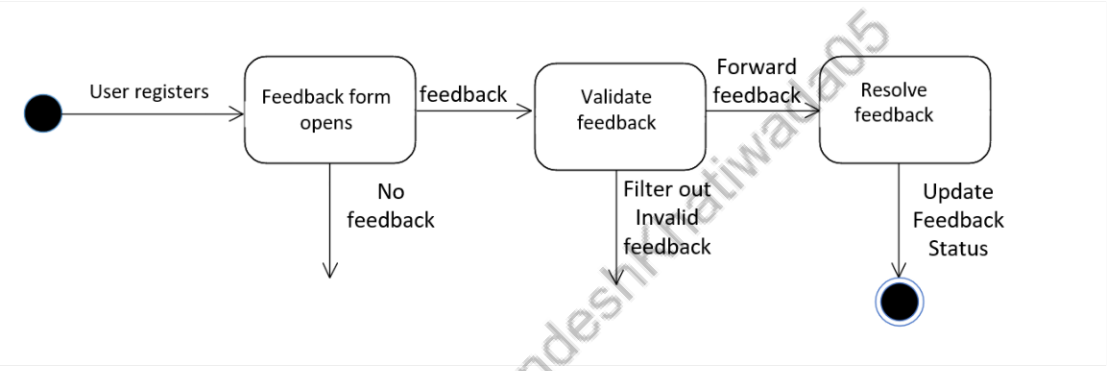


Figure 4.5: State machine diagram for Naagarik Feedback

2. Activity Diagrams

Activity Diagrams are used to show how control moves through a system. An activity diagram is another tool we may use to visualize the actions that go into carrying out a use case. Activity diagrams are used to model both concurrent and sequential activities. Thus, we essentially use an activity diagram to visually represent workflows. The condition of flow and the order in which it occurs are the main topics of an activity diagram. We utilize an activity diagram to explain or illustrate the source of a specific event.

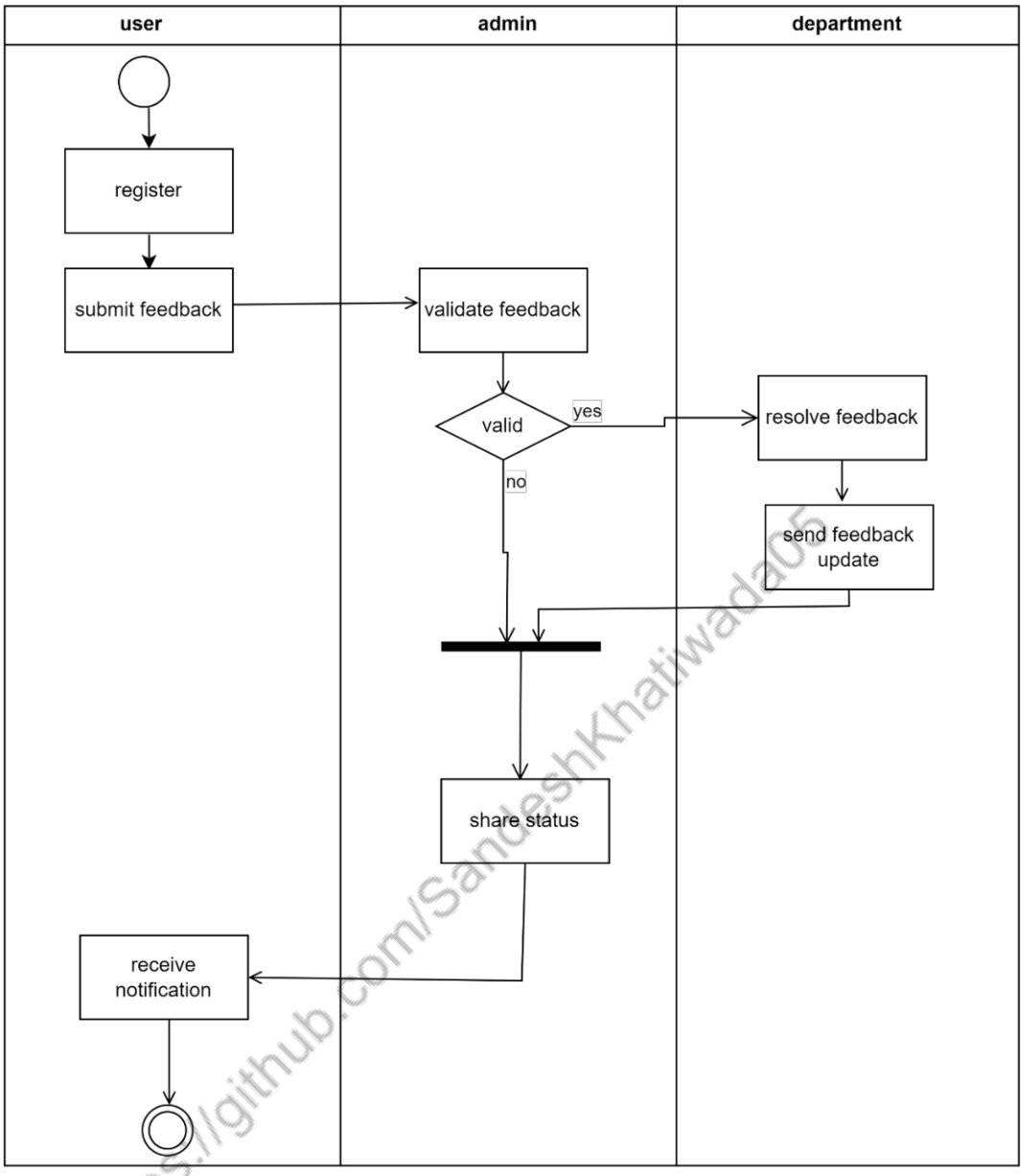


Figure 4.6: Activity diagram for Naagarik Feedback

3. Use Case Diagrams

Utilization Case Diagrams are used to show how a system or a component of a system operates. They are frequently used to show the system's functional requirements as well as how it interacts with outside agents, or actors. A use case is essentially a diagram that shows several circumstances in which the system might be applied. A use case diagram provides an overview of the functions of a system or a portion of a system without delving into implementation specifics.

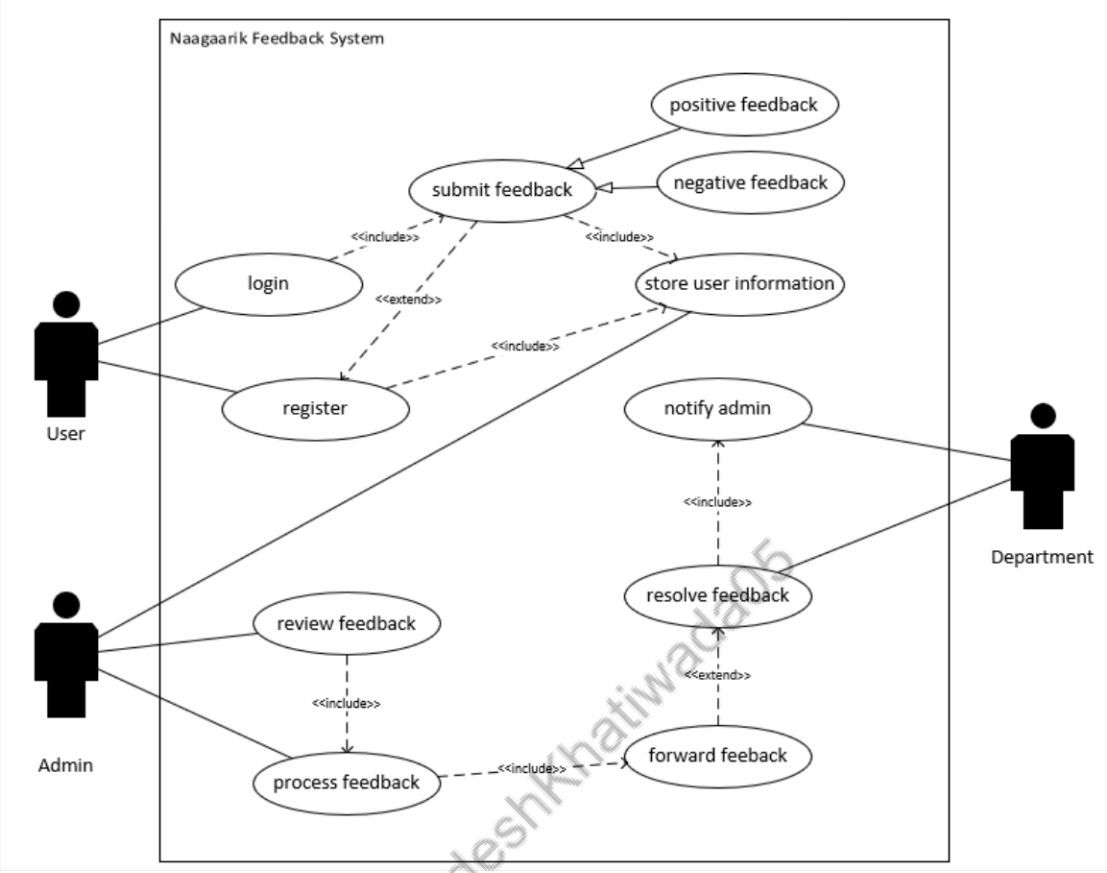


Figure 4.7: Use case diagram for submitting feedback

4. Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.

- We can also use the terms event diagrams or event scenarios to refer to a sequence diagram.
- Sequence diagrams describe how and in what order the objects in a system function.
- These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

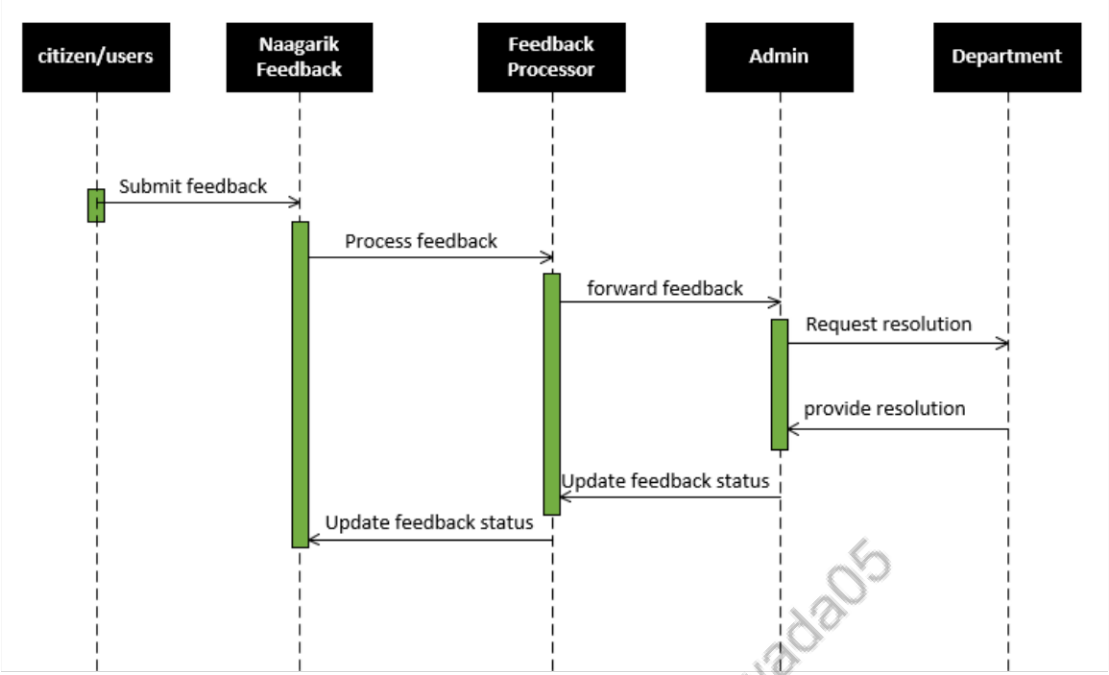


Figure 4.8: Sequence diagram for Naagarik Feedback