SOFTWARE ENGINEERING CONCEPTS
LAB MANUAL

DIVYA SURESH	(220701069)
GOPIKRISHNAN R	(220701075)
HARIESH A	(220701081)
HARISH RAGAVENDAR S	(220701087)
JAYA SHIVAANI B M	(220701099)

TABLE OF CONTENTS

OVERVIEW OF THE PROJECT

The **Repeated Faults Analysis** is a project which primarily focuses on a problem which is the high rate of incorrectly resolved customer complaints in the telecom sector, leading to repeated tickets and increased operational costs. This issue results in decreased customer satisfaction and higher expenses for the company.

The problem we are trying to solve is the high number of re-addressed customer complaints in the telecom industry. This happens when customer care representatives incorrectly close tickets.

The customer satisfaction can also be measured as NPS, Net Promoter Score (opposite group is called as Detractors). While companies try to increase their NPS scores by giving attractive offers and packages, they also try to listen their customer's issue and try to address them in the most effective way. While offering good telecom monthly plans is done by the bright and brilliant management minds, the heavy lifting of addressing customers issues falls completely on the customer care

operators. This also leads to an increased scope of errors while resolving the issues.

Proposed Solution

1. Main Interface:

Company Information: Brief overview of the company, its mission, and services offered.

Navigation Menu: Links to various sections of the website including support, plans, FAQs, and contact information.

Search Functionality: A prominently placed search bar allowing users to quickly find information.

Customer Issue Submission: A clearly visible button that directs users to the complaint registration interface.

2. Search Functionality

Keyword Search: Allows users to search for specific information using keywords.

Auto-suggestions: Offers suggestions as user type, helping them find relevant content quickly.

Filters: Provides options to filter search results by category (e.g., billing, technical support, plans).

3.Complaint Submission

Visibility: Strategically placed on the main interface and other relevant pages for easy access.

One-Click Access: Directs users immediately to the complaint registration interface with a single click.

4. Complaint Registration Interface

User-Friendly Interface: Intuitive design with clear instructions and easy-to-use interface.

- **Guided Complaint Registration:** A step-by-step process where users answer a few questions to diagnose and potentially resolve their issue.
- **Initial Questions:** Collect basic information about the problem (e.g., type of issue, affected service).

Self-service Troubleshooting: Based on user responses, provide possible solutions to common issues.

- **Advanced Complaint Registration:** If the problem is not resolved or is more complicated, users can register a detailed complaint.
- **Detailed Information Form:** Collects comprehensive information about the issue, including attachments if necessary (e.g., screenshots).

Ticket Generation: Automatically generates a ticket number for tracking the complaint.

Complaint Receival: Ensures the correct logging of complaints and their status in the system.

Status Tracking: Users can enter their ticket number to check the status of their complaint.

Feedback Mechanism: After the issue is resolved, users can provide feedback on the service received.

Benefits of the Proposed System

For Users:

- **Ease of Use:**

  Simplifies the process of raising complaints with a straightforward and intuitive interface.

- **Quick Resolution:**

  Helps users resolve common issues quickly by providing immediate answers based on their responses.

- **Tracking:**

  Enables users to track the status of their complaints with a ticket number, providing transparency and reassurance.

- **Reduced Frustration:**

  Minimizes the need for repeated contact with customer care by ensuring complaints are accurately recorded and tracked.

For the Company:

- **Efficiency:**

  Reduces the likelihood of incorrect ticket closures by guiding users through a structured complaint registration process.

- **Cost Savings:**

  Lowers operational costs by minimizing the number of re-addressed tickets and ensuring more efficient handling of complaints.

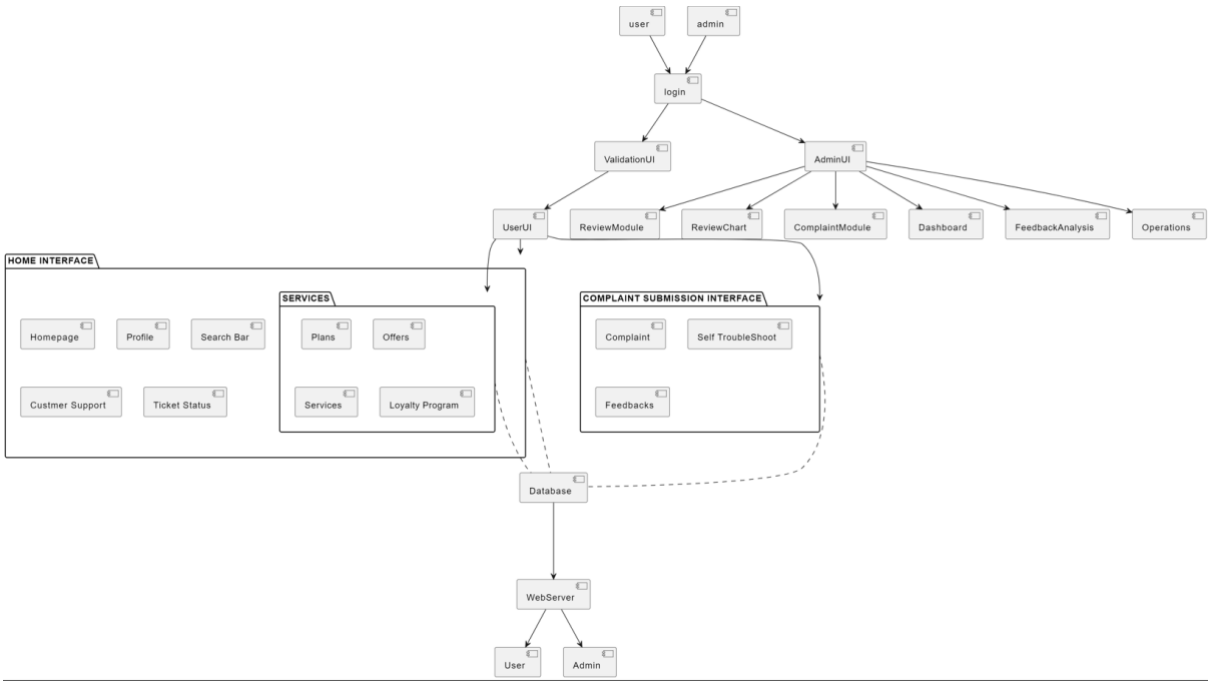
- **Customer Satisfaction:**

  Improves Net Promoter Score (NPS) by providing a more reliable and satisfying customer service experience.

- **Data Collection:**

Collects structured data on common issues and complaints, which can be used to identify areas for improvement and training for customer care representatives.

BUSINESS ARCHITECTURE



Key Components and Flow:

1. **Login and Validation**:
    - Users and administrators access the system through a **login** interface.
    - Both roles undergo validation via the **ValidationUI**.
2. **User Interface (UserUI)**:

- **Home Interface**: The main navigation area for users, including:
  - **Homepage**: Main landing page.
  - **Profile**: User account information.
  - **Search Bar**: Search functionality.
  - **Customer Support**: Access to support services.
  - **Ticket Status**: Check the status of submitted complaints or tickets.
- **Services**: Additional services provided, including:
  - **Plans**: Information on available plans.
  - **Offers**: Special offers and promotions.
  - **Services**: General services offered.
  - **Loyalty Program**: Details on loyalty rewards and benefits.
- **Complaint Submission Interface**: Specifically for handling complaints, consisting of:
  - **Complaint**: Submit new complaints.
  - **Self Troubleshoot**: Self-help options for resolving common issues.
  - **Feedback**: Provide feedback on services and resolutions.

3. **Admin Interface (AdminUI)**:

- **ReviewModule** and **ReviewChart**: Tools for administrators to review complaints and feedback.
- **ComplaintModule**: Manages and processes user complaints.

- **Dashboard**: Provides a high-level overview of system operations and statuses.
- **FeedbackAnalysis**: Analyzes user feedback for insights and improvements.
- **Operations**: Handles day-to-day operations and administrative tasks.

4. **Data Management**:
    - **Database**: Central repository for storing all data, including user information, complaints, and feedback.
    - **WebServer**: Facilitates communication between the database and the interfaces for both users and administrators.

REQUIREMENT AS USER STORIES

FUNCTIONAL REQUIREMENTS

User Story 1: Search Functionality

User Story:
As a user, I want to be able to search for recharge plans, customer support, and complaints in the search bar so that I can access them directly.

Acceptance Criteria:

- The search bar should be prominently displayed on the homepage.
- Users should be able to search for "recharge plans," "customer support," and "complaints."
- Search results should be relevant and lead directly to the respective sections.
- The search functionality should return results within 2 seconds.

Discussion Points:

- Placement and design of the search bar.
- Relevance and speed of search results.
- Integration with the backend to fetch relevant data.

Consensus:

- Agreed on the importance of fast and relevant search results.
- Ensured that the search bar will be tested for responsiveness on various devices.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 3
- Team Member 2: 5
- Team Member 3: 3
- Team Member 4: 5
- Team Member 5: 3
- **Final Estimate:** 5

User Story 2: Complaint Registration Interface

User Story:
As a user, I want to answer a few questions in the complaint interface so that I can self-troubleshoot my problems.

Acceptance Criteria:

- The complaint interface should present a series of troubleshooting questions.
- Users should be able to indicate if the issue is resolved or if further assistance is needed.
- The interface should be user-friendly and guide users step-by-step.

Discussion Points:

- Types of common issues to include in the questionnaire.
- User experience and flow of the troubleshooting process.

- Handling cases where issues are not resolved through self-troubleshooting.

Consensus:

- Agreed on creating a comprehensive list of common issues.
- Emphasized the need for a smooth and intuitive user experience.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 5
- Team Member 2: 3
- Team Member 3: 5
- Team Member 4: 3
- Team Member 5: 5
- **Final Estimate:** 5

User Story 3: Complaint History

User Story:

As a user, I want to visit my complaints so that I can see all the complaints I have ever raised.

Acceptance Criteria:

- A "My Complaints" section should be accessible from the user's account.
- Users should see a list of their complaints with details like date, issue, status, and resolution.
- The section should allow filtering and sorting by date and status.

Discussion Points:

- Security and privacy of complaint data.
- User interface for displaying complaint history.

- Filtering and sorting functionalities.

Consensus:

- Ensured the complaint data is secure and accessible only to the user.
- Agreed on the importance of a clean and intuitive interface for viewing complaint history.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 3
- Team Member 2: 3
- Team Member 3: 3
- Team Member 4: 5
- Team Member 5: 3
- **Final Estimate:** 3

User Story 4: Complaint Ticket Number

User Story:

As a user, I want to receive a ticket number for a complaint raised so that I can track the status of the complaint.

Acceptance Criteria:

- A unique ticket number should be generated upon complaint submission.
- Users should receive the ticket number via email and on the confirmation screen.
- Users should be able to use this ticket number to track the progress and status of their complaint.

Discussion Points:

- Generation and format of the ticket number.
- Notification methods for delivering the ticket number.

- Tracking system for complaint status.

Consensus:

- Decided on an alphanumeric format for ticket numbers.
- Agreed on sending ticket numbers via email and on-screen confirmation.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 2
- Team Member 2: 2
- Team Member 3: 3
- Team Member 4: 2
- Team Member 5: 3
- **Final Estimate:** 3

User Story 5: Chatbot Assistance

User Story:

As a user, I want a chatbot to assist me so that I can navigate through the website effortlessly.

Acceptance Criteria:

- The chatbot should be accessible from the homepage.
- It should be capable of answering common queries related to recharge plans, customer support, and complaints.
- The chatbot should provide guided navigation and direct links to relevant pages.
- The interface should be user-friendly and support text-based interaction.

Discussion Points:

- Integration of the chatbot with existing systems.

- Common queries and responses to program into the chatbot.
- User experience and feedback mechanism for the chatbot.

Consensus:

- Agreed on starting with a set of predefined common queries.
- Emphasized the importance of a seamless user experience.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 5
- Team Member 2: 5
- Team Member 3: 8
- Team Member 4: 5
- Team Member 5: 8
- **Final Estimate:** 8

User Story 6: User-Friendly Homepage

User Story:

As a user, I want the homepage to provide company details and easy navigation options so that I can quickly find what I need.

Acceptance Criteria:

- The homepage should prominently display key company information.
- Navigation options should be clearly labeled and easy to use.
- The design should be responsive and work well on both desktop and mobile devices.
- Important sections like recharge plans, customer support, and complaints should be easily accessible.

Discussion Points:

- Key information to be displayed on the homepage.
- Design elements for clear and easy navigation.
- Responsiveness and cross-device compatibility.

Consensus:

- Agreed on the need for a clean and informative homepage.
- Ensured responsiveness across various devices for a consistent user experience.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 3
- Team Member 2: 5
- Team Member 3: 3
- Team Member 4: 3
- Team Member 5: 5
- **Final Estimate:** 5

User Story 7: Efficient Issue Resolution

User Story:

As a user, I want my issues to be resolved efficiently so that I do not need to repeatedly contact customer support.

Acceptance Criteria:

- The complaint registration and resolution process should minimize the need for follow-ups.
- Users should receive timely updates about their complaint status.
- Solutions provided should address the core issue effectively.
- The system should track unresolved issues and prioritize their resolution.

Discussion Points:

- Mechanisms to ensure efficient issue resolution.
- Communication channels for timely updates.
- Prioritization of unresolved issues.

Consensus:

- Agreed on implementing an efficient tracking and notification system.
- Ensured the resolution process targets the core issues effectively.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 5
- Team Member 2: 8
- Team Member 3: 5
- Team Member 4: 8
- Team Member 5: 5
- **Final Estimate:** 8

User Story 8: Feedback Mechanism

User Story:

As a user, I want to provide feedback on the resolution process so that the company can improve its services.

Acceptance Criteria:

- After a complaint is resolved, users should receive a prompt to provide feedback.
- The feedback form should be short and focused on the resolution process.
- Users should have the option to rate their satisfaction and provide additional comments.
- The feedback should be reviewed by the company to improve service quality.

Discussion Points:

- Timing and method of feedback prompts.
- Design and content of the feedback form.
- Processes for reviewing and acting on feedback.

Consensus:

- Agreed on a short and effective feedback form.
- Ensured the feedback process is integrated into the resolution workflow.

Poker Planning Votes (1, 2, 3, 5, 8):

- Team Member 1: 3
- Team Member 2: 5
- Team Member 3: 3
- Team Member 4: 5
- Team Member 5: 3
- **Final Estimate:** 5

Note: In Poker Planning, as the number goes higher, it generally indicates greater complexity and effort required to complete the user story.

NON-FUNCTIONAL REQUIREMENTS (NFR)

1.Performance:

- The website should load quickly and respond to user actions promptly.
- The complaint registration process should be efficient and not time-consuming.
- Ticket tracking should provide real-time or near real-time updates on complaint status.

2.Availability:

- The system should be highly available with minimal downtime to ensure users can access services when needed.

3.Security:

- User data, including complaint details and ticket information, must be securely stored and protected from unauthorized access.

4.Scalability:

- The system should be able to handle an increasing number of users and complaints without compromising performance.

5.Usability:

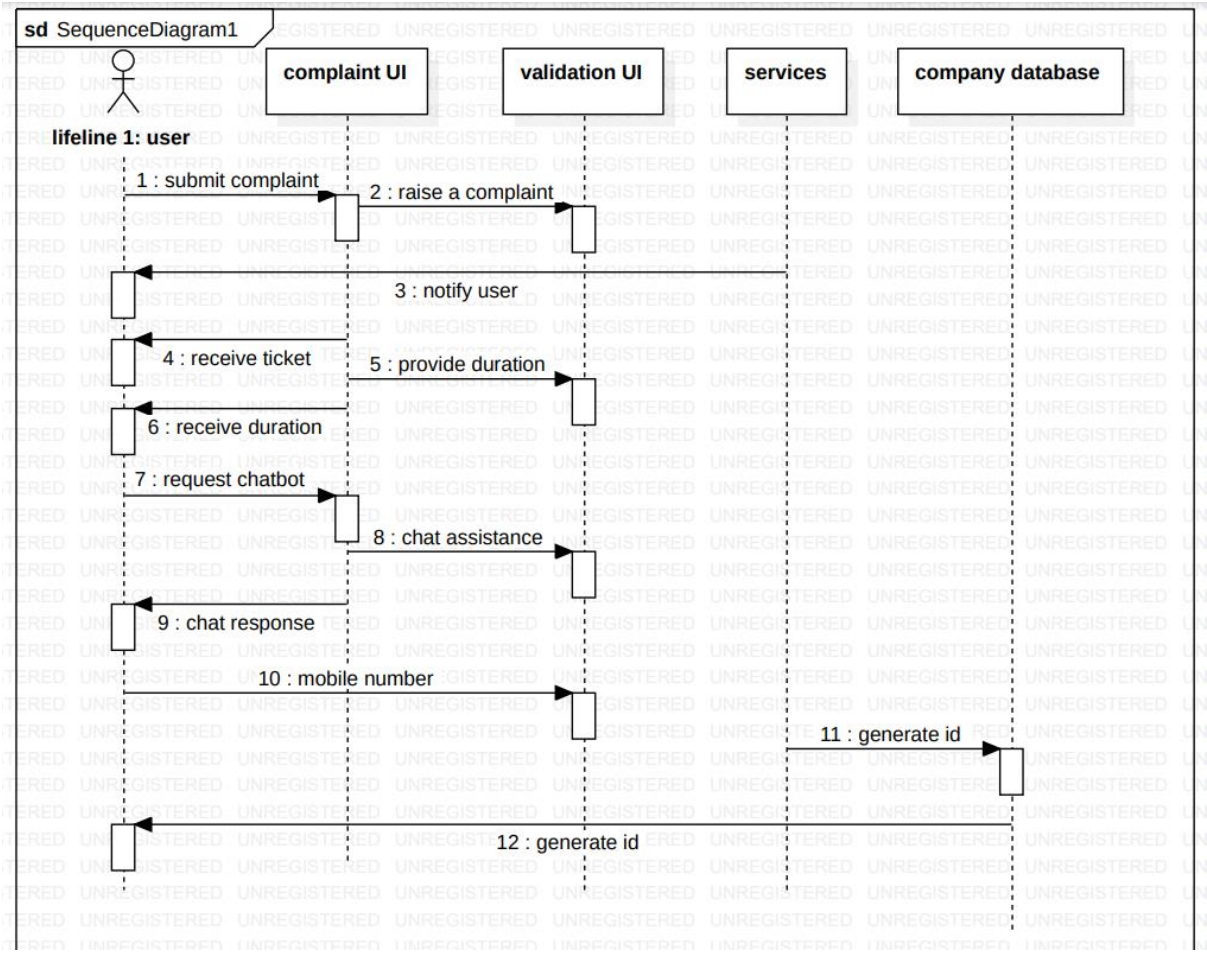
- The interface should be user-friendly and intuitive for users with varying technical skills.
- The complaint registration process should be clear and easy to follow.
- The system should be accessible to users with disabilities.

6.Maintainability:

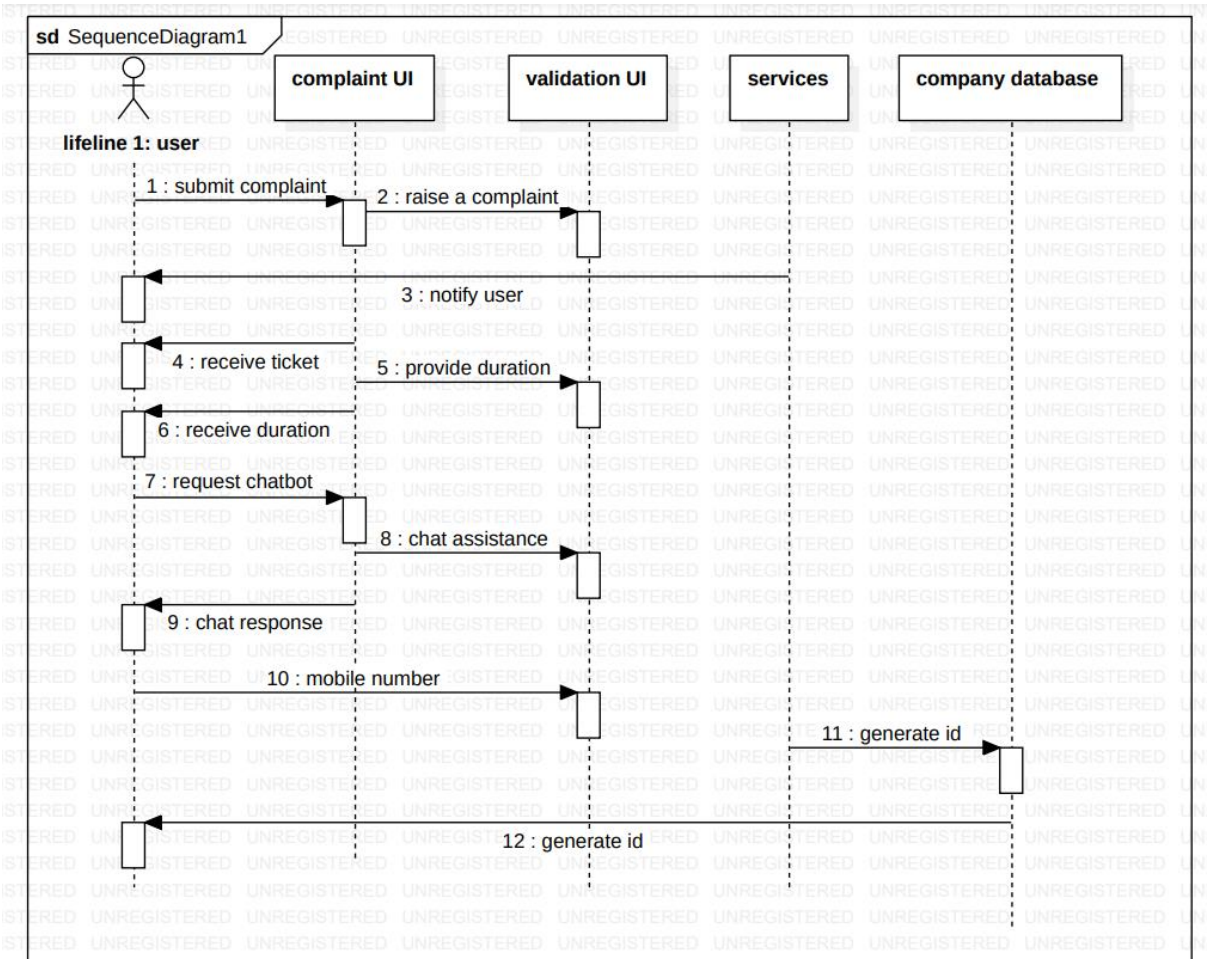
- The system should be designed with maintainability in mind, allowing for easy updates, bug fixes, and future enhancements.

SEQUENCE DIAGRAM AND CLASS DIAGRAMS

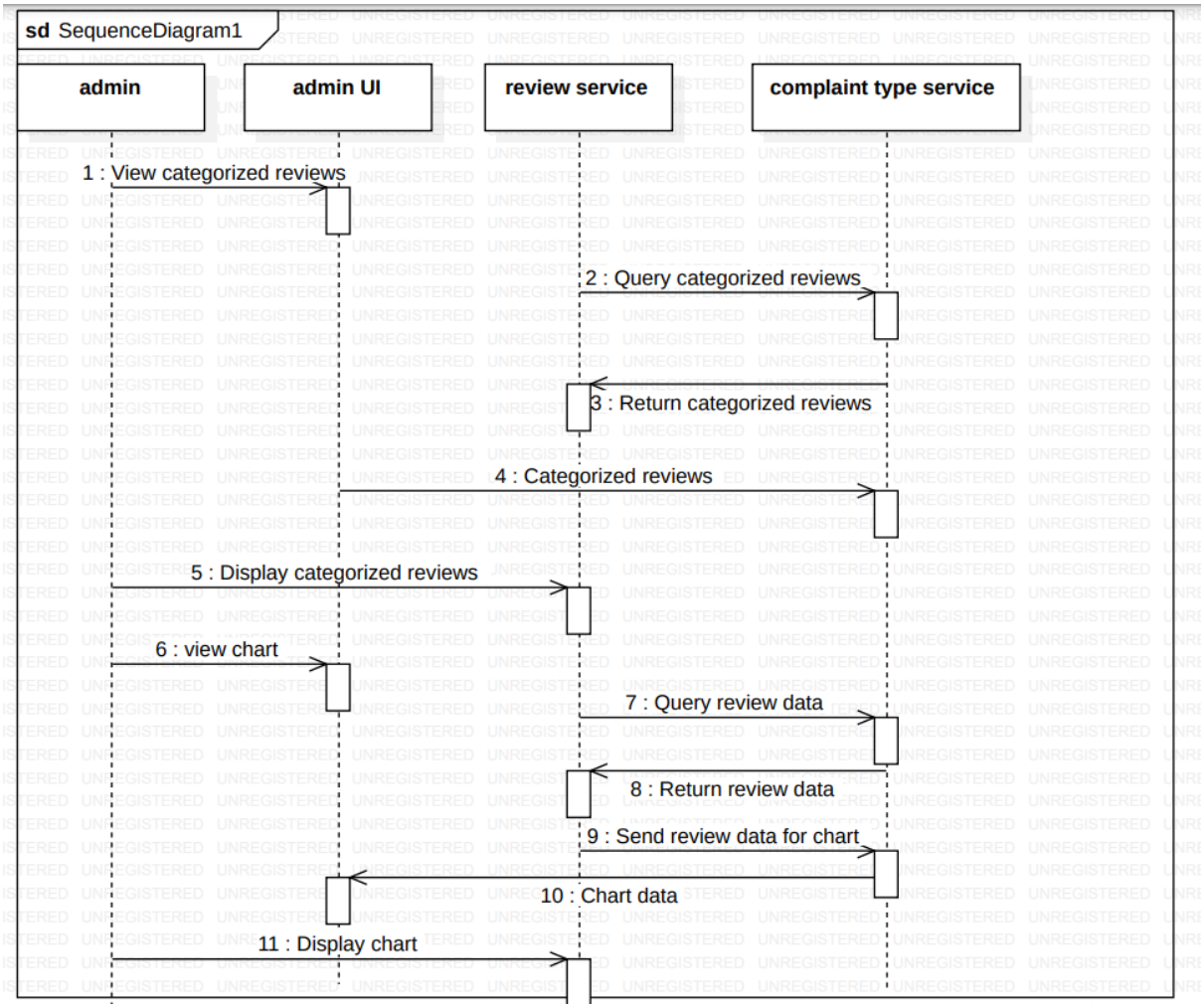
Sequence Diagram: Complaint Submission



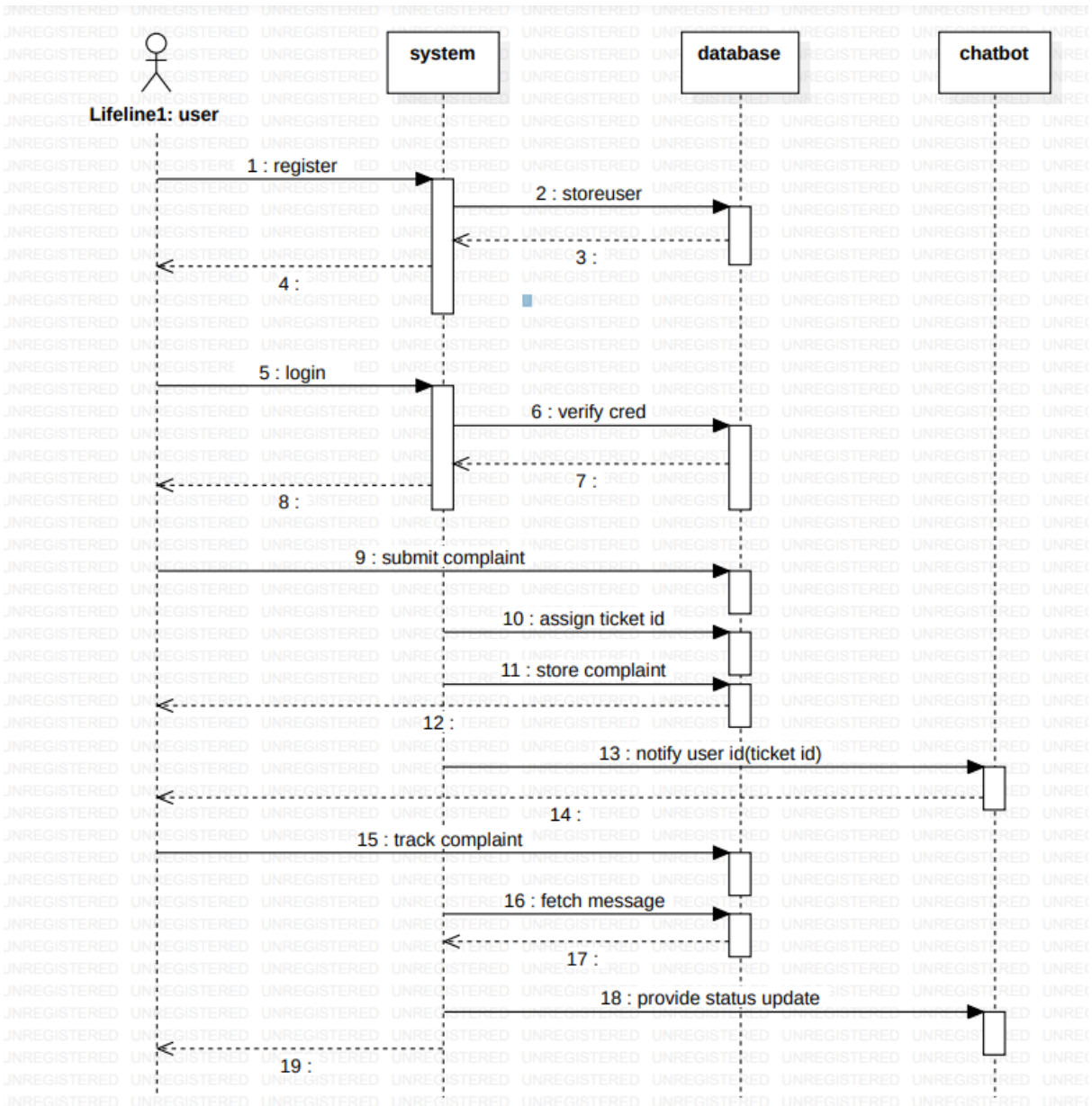
Sequence Diagram: Admin Database Management



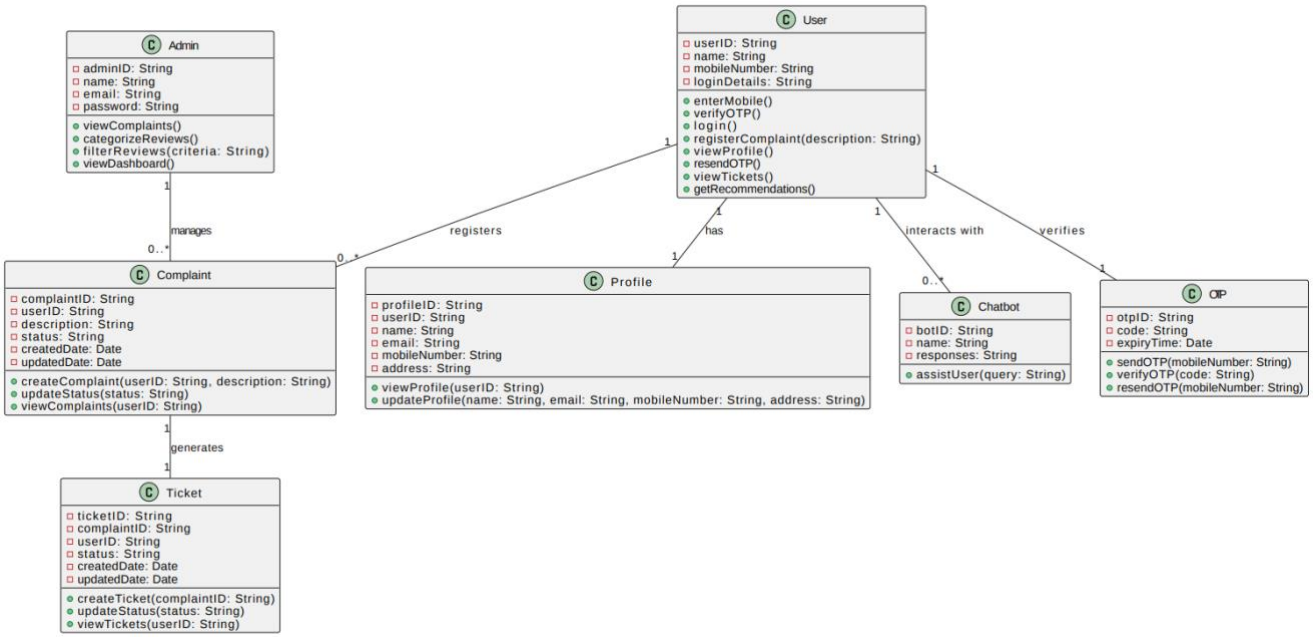
Sequence Diagram: Complaint Receival

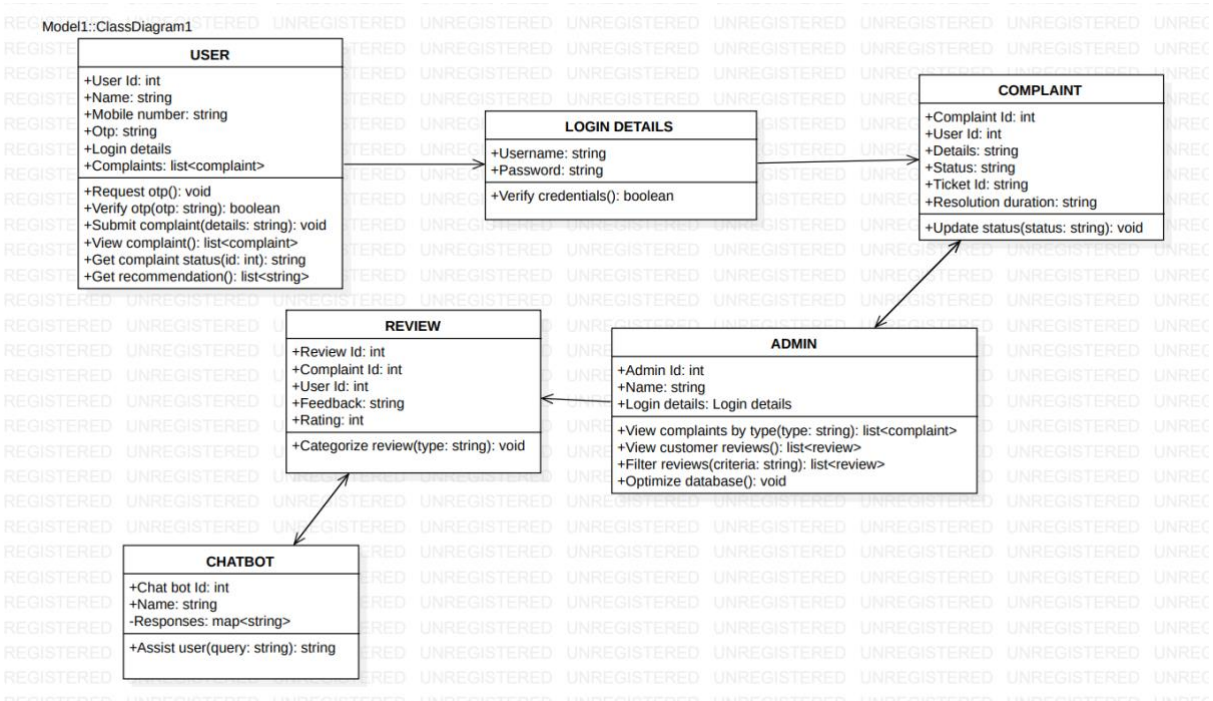


Sequence Diagram: Database Management

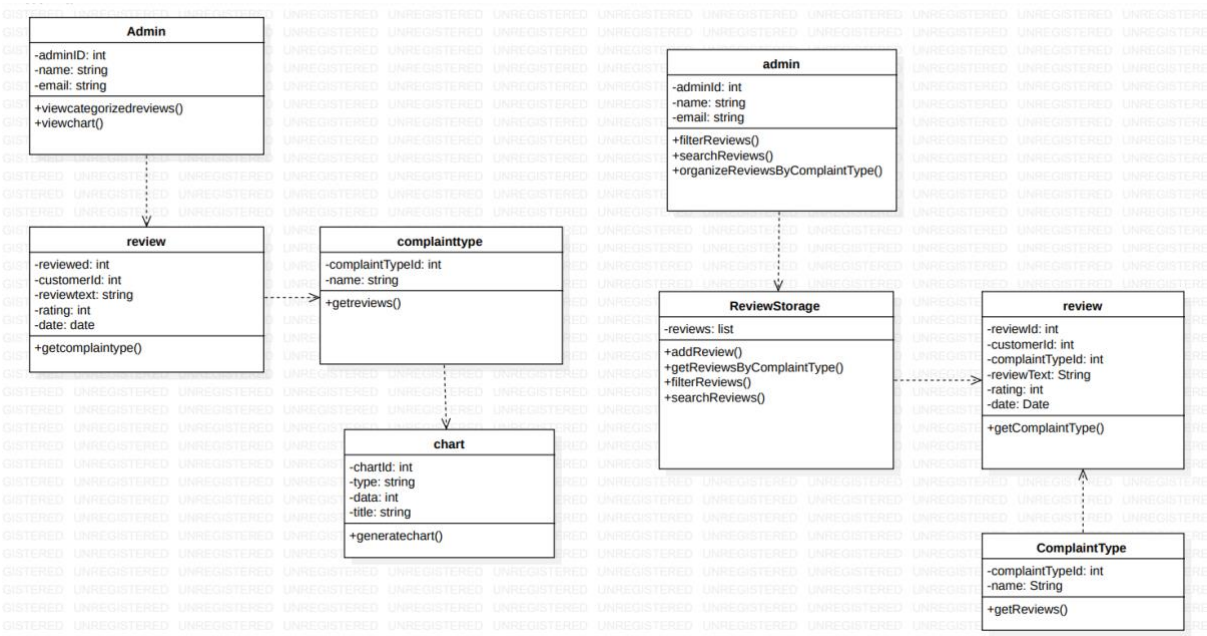


Class Diagram: System





Class Diagram: Database Server


Admin
-adminID: int
-name: string
-email: string
+viewcategorizedreviews()
+viewchart()
admin
-adminId: int
-name: string
-email: string
+filterReviews()
+searchReviews()
+organizeReviewsByComplaintType()
review
-reviewed: int
-customerId: int
-reviewtext: string
-rating: int
-date: date
+getcomplaintype()
complainttype
-complaintTypeId: int
-name: string
+getreviews()
ReviewStorage
-reviews: list
+addReview()
+getReviewsByComplaintType()
+filterReviews()
+searchReviews()
review
-reviewId: int
-customerId: int
-complaintTypeId: int
-reviewText: String
-rating: int
-date: Date
+getComplaintType()
chart
-chartId: int
-type: string
-data: int
-title: string
+generatechart()
ComplaintType
-complaintTypeId: int
-name: String
+getReviews()

ARCHITECTURE DIAGRAM DEPICTING SYSTEM

N-TIER ARCHITECTURE

1. Presentation Layer (Web Tier)

- **Components**:

  Homepage: Displays company details, enhancing transparency and trust.

  Search Bar: Facilitates easy navigation to recharge plans, customer support, and complaints.

  Complaint Button: Directs users to the complaint screen.

  Complaint Screen: User-friendly GUI for registering complaints with guided questions for self-troubleshooting and detailed complaint registration for complex issues.

  Ticket Tracking: Allows users to track the progress of their complaints using ticket numbers.

2. Business Logic Layer

- **Components**:

  UserService: Manages user authentication, registration, and profile updates.

  ComplaintService: Handles the creation, status updates, and resolution of complaints.

  TicketService: Generates and updates ticket statuses.

SearchService: Executes search queries and manages search history.

PlanService: Retrieves and manages telecom plans.

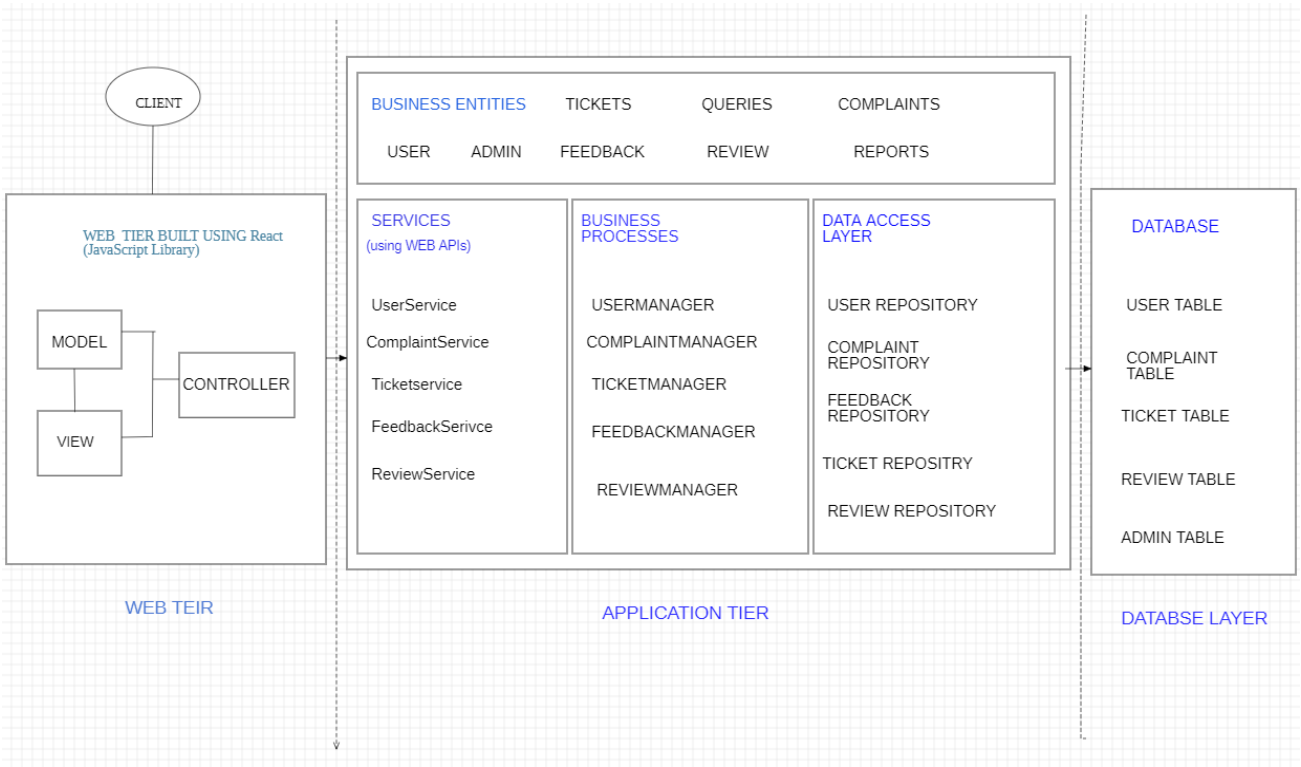
FeedbackService: Submits and analyzes user feedback.

NotificationService: Sends and updates notifications.

3. Data Access Layer

- **Components**:

  Database: Stores user data, complaints, tickets, search queries, plans, feedback, chatbot sessions, and notifications.



In an N-tier architecture, the web tier (or presentation layer) is responsible for managing the user interface and user experience. It follows the Model-View-Controller (MVC) pattern to separate concerns and improve maintainability.

Models

Models represent the data and business logic. They interact with services in the business logic layer to retrieve and manipulate data. Here are the models corresponding to our business entities:

1. **UserModel**
    - **Attributes**: userID, username, email, password, contactInfo, role
    - **Methods**: authenticate(), register(), updateProfile(), resetPassword()
2. **ComplaintModel**
    - **Attributes**: complaintID, userID, subject, description, status, priority, createdDate, resolvedDate
    - **Methods**: create(), updateStatus(), addResolution(), getProgress()
3. **TicketModel**
    - **Attributes**: ticketID, complaintID, userID, status, createdDate, lastUpdatedDate
    - **Methods**: generate(), updateStatus(), getStatus()
4. **SearchQueryModel**
    - **Attributes**: queryID, userID, queryText, queryDate, searchResults
    - **Methods**: execute(), saveHistory(), fetchResults()
5. **PlanModel**

- **Attributes**: planID, planName, planDetails, cost, validity, benefits
- **Methods**: getAll(), update(), delete(), create()

6. **FeedbackModel**
   - **Attributes**: feedbackID, userID, complaintID, rating, comments, feedbackDate
   - **Methods**: submit(), retrieve(), analyze()
7. **NotificationModel**
   - **Attributes**: notificationID, userID, message, sentDate, status
   - **Methods**: send(), updateStatus(), retrieve()

Views

Views are responsible for presenting data to the user and capturing user input. Here are the key views for our system:

1. **HomePageView**
   - Displays company details, a search bar, and navigation options.
   - Components: Search bar, company information section, navigation links.
2. **ComplaintFormView**
   - Presents a user-friendly GUI to register complaints.
   - Components: Form fields for subject, description, troubleshooting questions, and submit button.
3. **ComplaintHistoryView**
   - Displays all complaints raised by the user.
   - Components: List of complaints with status, dates, and links to details.

4. **ComplaintDetailsView**
    - Shows detailed information and progress of a specific complaint.
    - Components: Complaint details, status updates, resolution history, and feedback form.
5. **TicketStatusView**
    - Allows users to track the status of their complaint tickets.
    - Components: Input field for ticket number and display area for status updates.
6. **PlanListView**
    - Displays all available telecom plans.
    - Components: List of plans with details, costs, and benefits.
7. **FeedbackFormView**
    - Captures user feedback after a complaint is resolved.
    - Components: Rating stars, comment box, and submit button.

Controllers

Controllers handle the user input, process it, and update the model and view accordingly. Here are the controllers for our system:

1. **UserController**
    - **Methods**: login(), register(), updateProfile(), resetPassword()
    - **Responsibilities**: Handle user authentication, registration, profile updates, and password management.
2. **ComplaintController**
    - **Methods**: createComplaint(), updateComplaintStatus(), addResolution(), getComplaintProgress()

- **Responsibilities**: Manage complaint creation, status updates, and tracking.

3. **TicketController**
    - **Methods**: generateTicket(), updateTicketStatus(), getTicketStatus()
    - **Responsibilities**: Handle ticket generation, status updates, and tracking.
4. **SearchController**
    - **Methods**: executeSearch(), saveSearchHistory(), fetchSearchResults()
    - **Responsibilities**: Execute search queries and manage search history.
5. **PlanController**
    - **Methods**: getAllPlans(), updatePlan(), deletePlan(), createPlan()
    - **Responsibilities**: Manage telecom plans, including retrieval, updates, deletions, and creation.
6. **FeedbackController**
    - **Methods**: submitFeedback(), retrieveFeedback(), analyzeFeedback()
    - **Responsibilities**: Handle feedback submission and analysis.
7. **NotificationController**
    - **Methods**: sendNotification(), updateNotificationStatus(), retrieveNotifications()
    - **Responsibilities**: Handle sending, updating, and retrieving notifications.
    -

Example Interactions

1. **Submitting a Complaint**
    - **View**: User fills out the form in ComplaintFormView.
    - **Controller**: ComplaintController.createComplaint() processes the form input.
    - **Model**: ComplaintModel.create() saves the complaint data.
2. **Tracking Complaint Status**
    - **View**: User enters ticket number in TicketStatusView.
    - **Controller**: TicketController.getTicketStatus() retrieves the status.
    - **Model**: TicketModel.getStatus() fetches the status data.
    - **View**: Displays the status to the user.
3. **Searching for Plans**
    - **View**: User enters search text in the search bar on HomePageView.
    - **Controller**: SearchController.executeSearch() processes the search query.
    - **Model**: SearchQueryModel.execute() retrieves matching plans.
    - **View**: Displays search results to the user.

TEST STRATEGY

Objectives

- Ensure all functionalities meet specified requirements.
- Verify the system's usability and accessibility.
- Validate the accuracy and efficiency of complaint handling and ticket tracking.
- Test the integration of various system components.
- Ensure the system's performance, security, and scalability.

Test Environment

- **Development Environment**: For initial testing and debugging.
- **Staging Environment**: A replica of the production environment for final validation.
- **Production Environment**: For live monitoring and validation post-deployment.

Test Plan

Functional Testing

Objective: Verify that each function operates in conformance with the requirement specifications.

User Stories and Test Cases:

1. **Search Functionality**
    - *As a user, I want to be able to search for recharge plans, customer support, complaints in the search bar so that I can visit them directly.*
        - **Test Case 1.1**: Ensure the search bar returns relevant results for recharge plans.
        - **Test Case 1.2**: Ensure the search bar returns relevant results for customer support.
        - **Test Case 1.3**: Ensure the search bar returns relevant results for complaints.
2. **Complaint Registration**
    - *As a user, I should be able to answer a few questions in the complaint interface so that I can self-troubleshoot my problems.*
        - **Test Case 2.1**: Ensure users can see a series of questions related to their complaint.
        - **Test Case 2.2**: Ensure the system provides troubleshooting steps based on user answers.

3. **View Complaints**
    - *As a user, I want to visit my complaints so that I can see all the complaints I have ever raised.*
        - **Test Case 3.1**: Ensure users can view a list of all complaints they have raised.
        - **Test Case 3.2**: Ensure users can see the status and details of each complaint.
4. **Ticket Generation and Tracking**
    - *As a user, I should be able to receive a ticket number for a complaint raised so that I can track the status of complaint raised.*
        - **Test Case 4.1**: Ensure a ticket number is generated when a complaint is submitted.
        - **Test Case 4.2**: Ensure users can track the status of their complaints using the ticket number.

Integration Testing

Objective: Verify the interaction between different components and layers.

Test Cases:

- **API Testing**: Validate that APIs return correct data and handle errors gracefully.
- **Data Consistency**: Ensure data is consistent across layers and components.

Performance Testing

Objective: Ensure the system performs well under expected load conditions.

Test Cases:

- **Load Testing**: Ensure the system can handle the expected number of concurrent users.
- **Stress Testing**: Identify the system's breaking point under extreme conditions.
- **Scalability Testing**: Ensure the system can scale to accommodate increased load.

Security Testing

Objective: Ensure the system is secure from vulnerabilities and threats.

Test Cases:

- **Authentication and Authorization**: Ensure only authorized users can access sensitive data and functionalities.
- **Data Protection**: Ensure data is encrypted and protected from unauthorized access.
- **Vulnerability Scanning**: Identify and fix security vulnerabilities.

Usability Testing

Objective: Ensure the system is user-friendly and meets user expectations.

Test Cases:

- **User Interface**: Ensure the interface is intuitive and easy to navigate.
- **User Experience**: Validate that users can complete tasks efficiently and effectively.

Regression Testing

Objective: Ensure new changes do not adversely affect existing functionality.

Test Cases:

- **Re-run Functional Tests**: Ensure previously passed tests still pass after changes.
- **Impact Analysis**: Identify and test areas impacted by recent changes.

6. Test Deliverables

- Test Plan Document
- Test Cases and Test Scripts
- Test Execution Reports
- Defect Reports
- Final Test Summary Report

7. Entry and Exit Criteria

Entry Criteria:

- Requirements are finalized and approved.
- Test environment is set up.
- Test cases are reviewed and approved.

Exit Criteria:

- All test cases executed.
- No critical defects remain open.
- Test summary report is reviewed and approved.