Combine Task Team (CTT) Web Portal
Documentation

1. Introduction

Welcome to the comprehensive documentation for the **Combine Task Team (CTT) Web Portal**, a robust solution powered by **Claystone Tech**, created to streamline the resolution of internal complaints within DHA department. This documentation aims to provide an in-depth understanding of the **CTT Web Portal** and guide you through its effective use.

1.1 Purpose

The **Combine Task Team (CTT) Web Portal** is a comprehensive solution designed to streamline the resolution of internal complaints within the DHA department. This portal serves as a robust tool, integrating various features to facilitate efficient complaint management and enhance the department's operational effectiveness. Its primary objectives include:

Enhanced Security: The **CTT Web Portal** aims to improve the security of DHA operations by providing a centralized platform for handling complaints and issues promptly, ensuring a safe and secure environment for all stakeholders.

Efficient Grievance Resolution: An essential goal of the portal is to streamline the complaint resolution process. It offers a user-friendly interface for registering and tracking complaints, enabling quick and effective communication between departments and complainants.

Community Engagement: The portal fosters community engagement by involving users in the resolution process and providing transparency in addressing concerns. It enhances the sense of involvement and collaboration among DHA members.

Medical Services: The portal may extend its functionality to include features related to medical complaints and services within the DHA department, ensuring timely access to medical assistance when needed.

1.2 Scope

This documentation provides a comprehensive exploration of the **CTT Web Portal**, including its main functionalities, user interactions, and system architecture. It covers topics such as complaint registration, management, and priority handling. The scope of this documentation includes:

- Detailed instructions on using the **CTT Web Portal** effectively.
- Information on recent changes and updates as per DHA requirements.
- Troubleshooting guidelines for common issues.
- Frequently Asked Questions (FAQs) to assist users.
- Developer guidance for admin classes and API information.

1.3 Audience

This documentation is intended for users and stakeholders involved in the **CTT Web Portal**. It is valuable for both internal department users seeking to manage complaints efficiently and developers responsible for maintaining and enhancing the portal's functionality. Whether you are an internal user or an administrator overseeing the portal's operation, this documentation serves as a comprehensive resource to help you make the most of the **CTT Web Portal.**

2. Network Communication

Network Communication in the context of the CTT Web Portal refers to the processes and protocols involved in exchanging data and information between the portal and external servers or services over a network, typically the internet. Effective network communication is crucial for features like real-time updates, data synchronization, and accessing remote resources.

- **Protocol Selection:** The CTT Web Portal utilizes industry-standard communication protocols such as HTTPS for secure data exchange.
- **API Endpoints:** Defines and utilizes specific API endpoints for various functionalities, allowing the portal to send requests and receive responses from remote servers.
- **Data Serialization:** Data is serialized into the appropriate format (i.e., JSON) before sending requests and deserialized upon receiving responses.
- **Secure Communication:** Secure Socket Layer (SSL) or Transport Layer Security (TLS) is implemented to encrypt data transmitted between the portal and servers, ensuring data privacy and security.
- **Authentication:** Authentication mechanisms (i.e., API keys / tokens) are employed to verify the identity of the portal and users when interacting with external services.
- **Error Handling:** Robust error handling mechanisms are in place to manage network-related errors gracefully, providing informative error messages to users when necessary.
- **Network Monitoring:** Tools and libraries may be used to monitor network requests and responses, enabling performance analysis and issue detection.

2.1. RESTful API Integration

RESTful API Integration involves connecting the CTT Web Portal with external services or APIs that follow the principles of Representational State Transfer (REST). RESTful APIs are designed to be simple, scalable, and stateless, making them ideal for web-based portal integration.

- **Endpoint Discovery:** Developers identify and document the RESTful API endpoints provided by external services, understanding their functionality and data exchange format.
- **HTTPs Methods:** The portal uses standard HTTPs methods (i.e., GET, POST, UPDATE, DELETE) to interact with RESTful API endpoints, aligning requests with the intended actions (e.g., retrieving data, submitting data).
- **Request Construction:** Requests are constructed with the required headers, parameters, and payload data following the API documentation.
- **Response Handling:** Responses from the RESTful API are parsed and processed to extract relevant data and handle errors gracefully.
- **Authentication:** Authentication tokens and credentials are included in requests to ensure authorized access to protected API endpoints.
- **Pagination:** When dealing with large data sets, pagination techniques are implemented to efficiently retrieve and display data.
- **Rate Limiting:** If the RESTful API imposes rate limits, the portal respects these limits to avoid overloading the external service.
- **Error Codes:** Error codes and status messages from the API are translated into user-friendly messages and actions within the portal.
- **Testing and Debugging**: Extensive testing and debugging are conducted to ensure the correct integration of RESTful APIs, including handling edge cases and unexpected responses.
- **Versioning:** The portal accommodates API versioning to ensure compatibility with evolving API specifications.

3. Development Environment Setup:

- This section guides developers on setting up their development environment, including:

- **IDE Installation:** Instructions for installing an Integrated Development Environment (IDE) suitable for **CTT Web Portal** development.
- **Choosing an IDE:** Recommendations on selecting an appropriate IDE for the project.
- **Development Tools:** Here is a list of the tools employed in this project:
  - **VSCode (Visual Studio Code):** We utilize **VSCode** as our primary code editor for its versatility and extensive extension support.
  - **Django (Python Web Framework): Django** powers our web development, providing a solid foundation with features such as ORM, routing, and security.
  - **Microsoft Word (Documentation):** For document creation and formatting, we rely on Microsoft Word, an industry-standard word processing application.
  - **pip (Package Manager):** We use pip as our Python package manager, streamlining library and dependency management.
  - **pytest (Testing Framework):** pytest serves as our Python testing framework, enabling efficient unit testing.

  - **Web Browsers (e.g., Google Chrome, Firefox):** We prefer Google Chrome for testing web applications and sites due to its robust developer tools.
  - **Slack (Collaboration):** Slack enhances our team collaboration, providing channels, direct messaging, and integrations with various development tools.
- **Third-Party Libraries:** In our project, we have integrated third-party libraries to enhance its functionality and user experience. Specifically, we have incorporated JavaScript (JS) for dynamic interactivity and Highcharts CDN for creating interactive charts and visualizations. These libraries play a vital role in providing users with a seamless and visually appealing experience while interacting with our project.

4. Project Structure:

The project is structured as follows:

4.1: Main Directory (CTT): The top-level directory contains three main subdirectories:

- **project:** This directory houses the core project files and code.
- **venv:** Here resides the virtual environment used for managing project dependencies.
- **versions:** This directory may store different versions or backups of the project.

4.2: Project Directory (inside 'project'): Within the 'project' directory, you'll find several subdirectories:

- **app_admin:** This directory likely contains the main application code and includes administrative-related code and resources.
- **templates:** Here, templates for rendering web pages are stored.
- **assets:** This directory could house static assets like images, stylesheets, and JavaScript files, among others.

5. Architecture:

5.1 Design Patterns

Let me explain how the MVC/MVT pattern is applied in the context of the "CTT" project:

5.1. Model (M):

- In the CTT project, the "Model" represents the data structure and the database schema.

- Django provides an Object-Relational Mapping (ORM) system that defines models as Python classes. Each model class corresponds to a database table, and its attributes represent table fields.
- For example, if the project deals with user data, there might be a "User" model that defines the structure of the "users" table in the database.

5.2. View (V):

- The "View" in Django is responsible for handling user requests, processing data, and returning appropriate responses.
- Views in Django are Python functions or classes that take web requests and return web responses.
- In the CTT project, views are created to manage different aspects of the web application, such as rendering templates, processing form data, and interacting with the models.
- For instance, a view might render a user's profile page, retrieving data from the "User" model.

5.3. Template (T):

- In the MVC pattern, the "Controller" is responsible for handling user input and deciding which view to display. In Django, this role is combined with the "View."
- Django uses templates to separate the presentation layer from the logic layer.
- Templates are HTML files with placeholders for dynamic content. These placeholders are filled in with data from the views.
- In the CTT project, templates define the structure of web pages and how data from the views is presented to the user. For instance, an HTML template could define the layout of a user registration form.

6. Testing:

Testing is a crucial phase in the development lifecycle of the CTT Web Portal. It ensures that the portal functions as expected and delivers a reliable experience to users. The testing phase comprises two key components:

6.1. Unit Testing:

Unit testing is a fundamental testing method that evaluates individual components or units of the codebase in isolation. These units can be functions, classes, or modules that perform specific tasks within the application.

The primary purpose of unit testing is to validate the correctness of each isolated component. It focuses on ensuring that these components perform their intended functions accurately and consistently. Unit tests are typically written by developers to verify that their code functions as expected.

6.2. Key Aspects of Unit Testing:

- **Isolation:** Each unit is tested in isolation, meaning that external dependencies or interactions with other parts of the code are minimized or replaced with controlled inputs.
- **Automation:** Unit tests are automated to ensure that they can be executed repeatedly and consistently, making them a valuable tool for detecting regressions during development.
- **Assertions:** Unit tests include assertions that check whether the unit's output matches the expected results for a given set of inputs.
- **Coverage:** Developers aim to achieve high code coverage by writing tests that exercise different code paths, ensuring that a wide range of scenarios are tested.

6.3. Front-end Unit Testing:

For the front-end of the CTT Web Portal, unit tests may focus on components such as user interface elements, forms, and client-side logic. These tests validate that the user interface functions correctly and that user interactions produce the expected outcomes.

6.4. Back-end Unit Testing:

In the back-end, unit tests target individual functions, methods, or services responsible for processing data, business logic, and application rules. These tests ensure that each component of the server-side code performs its tasks accurately.

6.5. Integration Testing:

Integration testing assesses how different components or units of the CTT Web Portal interact with each other and with external systems, including databases, APIs, and third-party services. Unlike unit tests, which focus on isolated units, integration tests evaluate the collaboration between these components.

Integration testing aims to identify and address issues related to the interactions between different parts of the application. It ensures that data flows correctly between components and that the integration points are robust and reliable.

6.6. Key Aspects of Integration Testing:

- **Interactions:** Integration tests examine how various components collaborate, including the front-end and back-end, databases, and external APIs.
- **Data Flow:** These tests verify that data is correctly passed between different layers of the application and that it is processed accurately.
- **Dependency Testing:** Integration tests identify and validate dependencies on external systems, helping to uncover potential issues early in the development process.
- **End-to-End Scenarios:** In some cases, integration tests may encompass end-to-end scenarios that simulate user interactions with the entire system to ensure its holistic functionality.

6.7. Front-end Integration Testing:

Front-end integration tests validate how different parts of the user interface interact with one another, such as form submissions, navigation, and the presentation of data from the back-end.

6.8. Back-end Integration Testing:

In the back-end, integration tests focus on verifying that components like APIs, databases, and services work harmoniously. This includes testing API endpoints and their interactions with the database.

6.9. API Integration Testing:

Specifically for API integration testing, the goal is to validate that the CTT Web Portal can effectively communicate with external APIs. It checks the correctness of data exchange and the handling of API responses.

By conducting both unit testing and integration testing, the development team ensures the front-end, back-end, and API components of the CTT Web Portal work together seamlessly, delivering a reliable and functional user experience.

7. Documentation and Code Comments

Documentation and Code Comments are vital practices in software development for the CTT Web Portal. They involve providing comprehensive explanations, descriptions, and comments within the codebase. This helps developers, maintainers, and collaborators understand the code's functionality, purpose, and usage.

- **In-line Comments:** Meaningful comments are added throughout the codebase to explain complex logic, algorithms, or any non-trivial code segments.
- **Function and Method Documentation:** Each function or method includes a comment block describing its purpose, input parameters, return values, and usage examples.
- **Class and Module Documentation:** Documentation is provided at the class or module level, explaining the role and responsibilities of the class, its attributes, and how it fits into the overall system.
- **Usage Examples:** Code comments may include usage examples to illustrate how to use specific functions, methods, or classes effectively.
- **Code Guidelines:** Comments adhere to a consistent style and follow established coding guidelines for clarity and readability.
- **Change Logs:** Documentation may include change logs that track modifications, enhancements, and bug fixes for each code segment.
- **Dependency Documentation:** External dependencies and libraries used in the CTT Web Portal are documented, including version information and usage instructions.

8. Code Documentation Standards

Code Documentation Standards refer to the conventions and guidelines established for documenting code within the CTT Web Portal. These standards ensure consistency and readability across the codebase.

- **Comment Style:** Define a consistent comment style, such as using a specific syntax for in-line comments (e.g., "//" for single-line comments, "/* */" for multi-line comments).
- **Documentation Blocks:** Specify the format for documentation blocks, including what information to include (e.g., function descriptions, parameters, return values).

- **Naming Conventions:** Establish conventions for naming variables, functions, classes, and modules, ensuring that names are descriptive and meaningful.
- **Code Organization:** Define how code should be organized, including the placement of comments within the code structure.
- **Documenting Edge Cases:** Guidelines for documenting edge cases, error handling, and exceptional conditions to ensure robustness and understanding code behavior.
- **Version Control Integration:** Encourage inclusion of code documentation in version control commits and pull requests to maintain up-to-date documentation.

8.1. API Endpoint Documentation

API Endpoint Documentation is crucial for the CTT Web Portal, as it involves describing the endpoints and functionalities exposed by the portal's backend APIs. API documentation helps developers, both internal and external, understand how to interact with the portal's services programmatically.

- **Endpoint Descriptions:** Each API endpoint is documented with a clear and concise description of its purpose and expected behavior.
- **Request and Response Formats:** Documentation specifies the expected request formats (e.g., JSON, XML) and provides examples. It also defines the structure of the response and its possible status codes.
- **Authentication and Authorization:** Explain how authentication and authorization are handled for API access, including required tokens or credentials.
- **Endpoint Usage Examples:** Include practical usage examples for each endpoint, showcasing how to make requests and interpret responses.
- **Rate Limiting:** If applicable, communicate rate limiting policies to ensure fair usage of the API.
- **Error Handling:** Document error responses and their meanings, helping developers troubleshoot issues effectively.
- **Versioning:** If multiple API versions are supported, clarify how versioning is implemented and how developers can specify the desired version.
- **Change Log:** Maintain a change log for API endpoints, documenting updates, additions, and deprecations to keep developers informed about changes.

These revisions adapt the text to the CTT project documentation while retaining the key principles of code documentation and API endpoint documentation.

9. Main Dashboard

9.1. Dashboard Overview: Upon **logging in**, you will be presented with the main dashboard, which offers an informative overview of the complaint resolution process. The dashboard includes:

- The total number of complaints is categorized by type (e.g., electrical, security).
- Sub-category-wise complaint statistics.
- Area-wise complaint distribution, allowing you to identify areas with frequent issues easily.

10. Using the CTT Web Portal

➢ Viewing and Managing Complaints

1. Viewing Complaints: Access your registered complaints from the dashboard. Clicking on a specific complaint provides a detailed view, including its status (either 'Pending' or 'Resolved').

2. Managing Complaints: The portal's intuitive interface ensures you can efficiently identify and review your complaints for necessary action.

➢ Tagging and Referring Complaints

3. Tagging Departments: Internal department users can efficiently manage complaints requiring their expertise. You can:

- Tag specific departments relevant to a complaint that needs attention.
- Add essential information, including attachments and comments, to provide context and streamline the resolution process.

➢ Handling Complaints on Priority

4. **Priority Handling:** Complaints tagged for specific departments are given a higher priority, ensuring they receive immediate attention and resolution.

11. Recent Changes (DHA Requirements)

Recent Changes: These changes were implemented recently as per **DHA** requirements, based on feedback from the Combined Task Teams (beta) Feedback. Key changes include:

Upon logging in, users are now redirected to the "Complaints" page, or the current page is modified.

- The button text "Change" has been updated to "View."
- The "Recent Actions" box has been removed.
- The graph now displays total Phase-wise complaints with a drilldown option.
- Complaints are now presented in alphabetical order.
- The text below the table has been changed to "Complaints."
- Complaints are loaded in descending order of Complaint ID or Created On.
- Href tags on column headers have been removed, and clicking on headers now allows sorting.
- Pagination has been simplified, displaying all data on one page.
- The page selector button (1) on the Map page has been removed, and last month's data is loaded by default.
- Existing date filters (2) are used for customization.
- Duplicated text at point 4 has been corrected to "1234 Complaints."

12. Support: In the rare event that you encounter issues related to registration, login, or complaint management, our dedicated customer support team is ready to assist you promptly. Reach out to our support team for quick resolutions.

13. Frequently Asked Questions

1. How can I view the status of my complaints?

- ❖ After logging in, you will see your complaints on the main dashboard. Click on a complaint to view its details and status.

2. What should I include when tagging a department for a complaint?

- ❖ When tagging a department, provide clear information about the issue, any attachments, and comments to help the department understand and address the problem effectively.

3. How can I escalate a complaint to a higher priority?

- ❖ Complaints tagged for specific departments are automatically treated as a high priority. Ensure you tag the appropriate department to prioritize resolution.

4. What are the legal terms and conditions of using the CTT Web Portal?

- ❖ Please review the legal disclaimers, copyright information, and terms of use in the 'Legal Information' section of this documentation.

14. Developer Guidance

1. USER SIDE

- **Argument For Admin Classes -> admin.ModelAdmin**

1.1: Dashboard: classname DashboardAdmin takes model_admin as argument.

- template: complaint_dashboard_map.html
- Highcharts.js library is used for rendering graphs

1.2: Complaint: classname ComplaintAdmin takes model_admin as argument.

- **template:** default changelist_template.html (Complaint Grid) builtin.

1.3: Complaint Map: class ComplaintMapAdmin takes model_admin as argument.

- template: admin_templates/change_list_map.html.
- Map: OpenStreet Map CDN Based

1.4: QA Checklist: classname QAChecklistAdmin takes model_admin as argument.

- template -> admin_templates/change_form_template.html

1.5: Complaint: classname ComplaintAdmin takes model_admin as argument.

2. ADMIN SIDE

2.1: GEO INFO MODULE

2.1.1 Country: classname CountryAdmin takes model_admin as argument.

- template: default changelist_template.html.

2.1.2 State: classname StateAdmin takes model_admin as argument.

- template: default changelist_template.html.

2.1.3 City: class CityAdmin takes model_admin as argument.

- template: default changelist_template.html.

2.2: CORE SYSTEM MODULE:

2.2.1 AppMenu: classname AppMenuAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **menus**)

2.2.2 AppVersion: classname AppVersionAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **app version**)

2.2.3 Gender: class GenderAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **gender**)

2.2.4 NationalIdentityType: class NationalIdentityTypeAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **national identity type (cnic,nicop,poc)**)

2.2.5 Role: class RoleAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **User role and mdminrole**)

2.2.6 SpouseType: class SpouseTypeAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **spouse**)

2.2.7 SuperUser: class SuperUserAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **super**)

2.2.8 SystemStatus: class SystemStatusAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **user status, complaint status**)

2.2.9 SystemType: class SystemTypeAdmin takes model_admin as argument.

- template: default changelist_template.html.

  (This section includes details about **project categories like complaint types, user types**)

2.2.10 SystemUser: class SystemUserAdmin takes model_admin as argument.

- template: default changelist_template.html.

(This section includes details about the list of **system users** which are registered in a system)

2.2.11 UserType: class UserTypeAdmin takes model_admin as argument.

- template: default changelist_template.html.

(This section includes details about user type **admin or user**)

14. API Information

List of APIs

1. **APIs for Data Exchange**: Below is a list of APIs used in both the web portal and mobile app application for the exchange of information. These APIs enable seamless data transfer between the two platforms, ensuring synchronized complaint management and resolution.

New API :

- https://ctt.dhai-r.com.pkloginloginstatuses/qa_checklist_categories
- https://ctt.dhai-r.com.pkloginloginstatuses/qa_checklist_rating_choices
- https://ctt.dhai-r.com.pkloginloginstatuses/insert_qa_checklist
- https://ctt.dhai-r.com.pkloginloginstatuses/insert_qa_checklist

Old API:

- https://ctt.dhai-r.com.pkloginloginstatuses/categories
- https://ctt.dhai-r.com.pkloginloginstatuses/area_hierarchy
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint_statuses
- https://ctt.dhai-r.com.pkloginloginstatuses/insert_complaint
- https://ctt.dhai-r.com.pkloginloginstatuses/update_complaint
- https://ctt.dhai-r.com.pkloginloginstatuses/logout
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_cancel
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_feedback
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_resolved
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_close
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_priority
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_reopen
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_update
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint_list
- https://ctt.dhai-r.com.pkloginloginstatuses/own_complaint_list
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_membership
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/{id}
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint_action_comments/{id}
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint_action_files/{id}
- https://ctt.dhai-r.com.pkloginloginstatuses/getappversion

- https://ctt.dhai-r.com.pkloginloginstatuses/dashboard/emergencycontact
- https://ctt.dhai-r.com.pkloginloginstatuses/dashboard_api
- https://ctt.dhai-r.com.pkloginloginstatuses/complaint/complaint_fileupload

15. Project Requirements:

- **1. Technology Stack:**
  - Django
  - Bootstrap
  - HTML
  - CSS
  - JavaScript
  - jQuery

- **2. System Requirements:**
  - Operating Systems: Windows 7, 8, 10, or 11

- **3. Browser Compatibility:**
  - Chrome
  - Firefox
  - Edge
  - Safari
  - Opera

16. Legal Information

16.1 Disclaimers and Copyrights

1. Legal Disclaimer: Please be aware that **CTT (Combine Task Team)** is powered by **Claystone Tech**. All legal disclaimers, copyright information, and terms of use related to **Claystone Tech** apply to this web portal. Ensure that you review and understand these legal terms.

17. Conclusion

The **CTT Web Portal**, powered by **Claystone Tech**, offers a robust and user-friendly platform to manage internal complaints efficiently. This comprehensive documentation serves as a valuable resource to help you maximize the potential of the portal. Should you have any questions or require assistance beyond the FAQ section, please consult our dedicated support team.
Thank you for choosing **CTT** for your complaint resolution needs.

18: Appendix:

19: Glossary:

20: Reference:

21: Contact Information: