**Software Project Issue Tracker**

Project Documentation

**Comprehensive Installation and Setup Instructions**

This guide provides a detailed walkthrough for installing and setting up the **Software Project Issue Tracker**, a Django-based web application. Follow these instructions to configure the application for local development or production use.

**1. Prerequisites**

Ensure your system meets the following requirements before proceeding:

* **Python**: Version 3.8 or higher should be installed and accessible from the command line.
* **PostgreSQL**: Version 12 or higher should be installed and running for database management.
* **Git**: Required for cloning the project repository and version control.
* **Virtual Environment Tool**: Use venv or virtualenv to isolate project dependencies.
* **Node.js** (Optional): Required if the project includes a custom frontend using JavaScript frameworks like React or Vue.js.

**2. Installation Steps**

**Step 1: Clone the Repository**

* Navigate to the directory where you want to store the project.
* Clone the repository using Git to download the application source code.

**Step 2: Create a Virtual Environment**

* Create a virtual environment to isolate Python dependencies for this project.
* Activate the virtual environment:
  + On Linux/macOS: Use the source command.
  + On Windows: Run the activation script provided in the virtual environment folder.

**Step 3: Install Project Dependencies**

* Use the package manager pip to install all required Python libraries listed in the requirements.txt file. This ensures the application has all necessary dependencies (e.g., Django, database drivers).

**Step 4: Configure the Database**

* **Create a PostgreSQL Database**:
  + Set up a new database specifically for this project.
  + Create a database user with a strong password and grant the user full access to the database.
* **Update Django Settings**:
  + Open the settings.py file and update the DATABASES configuration.

**Step 5: Apply Database Migrations**

* Run Django’s migration commands to initialize the database schema:
  + This will create all necessary tables and relationships required by the application.

**Step 6: Create a Superuser**

* Create an administrator account using Django’s built-in command.
* Provide a username, email address, and password when prompted.

**Step 7: Collect Static Files**

* Run the collectstatic command, which collects all CSS, JavaScript, and image files into one directory called a static file. This is necessary to serve static files in production.

**Step 8: Starting the Development Server**

* Launch the Django development server to test the application locally.
* Open your web browser and navigate to the server’s default address (http://127.0.0.1:8000) to verify that the application is running.

**3. Deployment Instructions**

Additional steps for deploying to production:

**Step 1: Use a Production WSGI Server**

* Replace the development server of Django with Gunicorn or uWSGI. These are powerful production-grade WSGI servers, which are specially meant to handle production traffic effectively.

**Step 2: Configure a Reverse Proxy**

* To do that you need to have a reverse proxy server like; Nginx, Apache or lighttpd among others to proxy every request to the WSGI server.
* Configure the reverse proxy to serve static files and manage HTTPS connections.

**Step 3: Secure the Application**

* Obtain and configure SSL/TLS certificates to enable HTTPS.
* Store sensitive data, such as database credentials and secret keys, in environment variables instead of hardcoding them in the project files.

**Step 4: Optimize for Production**

* Disable Django’s debug mode by setting DEBUG = False in the settings.py file.
* Restrict access to the application by configuring the ALLOWED\_HOSTS setting.
* Implement caching mechanisms, such as Redis or Memcached, to improve performance.

**Step 5: Containerize the Application (Optional)**

* Use Docker to containerize the application for consistent deployment across different environments.
* Create a Docker image that includes the application code, dependencies, and configurations.

**4. Additional Notes**

**Common Issues and Troubleshooting**

* **Database Connection Errors**:
  + Ensure PostgreSQL is running and accessible.
  + Verify that the database credentials in the settings.py file match the database configuration.
* **Missing Dependencies**:
  + Re-run the dependency installation command to ensure all required packages are installed.
* **Static Files Not Loading**:
  + Verify that the static files were collected and that the static file directory is correctly configured in the production server.

**Development vs. Production**

* The development server is suitable for testing and debugging but should never be used in a production environment.
* Always configure a production-grade WSGI server and follow best practices for deployment.

By following these steps, you will have a fully functional Django-based **Software Project Issue Tracker** running in your local or production environment.

**User Manual: How to Use the Application**

This user manual provides detailed instructions on how to use the **Software Project Issue Tracker** application. The application supports multiple user roles, each with specific responsibilities and permissions. The roles include **Administrator**, **Project Manager**, **Developer**, and **Client/User**. Below is a step-by-step guide for each role, explaining how to navigate and perform their respective tasks within the application.

**1. Administrator Role**

The Administrator is responsible for managing the overall system, including user accounts, permissions, and application configurations.

**Key Responsibilities**

* Manage user accounts and roles.
* Oversee system configurations and settings.
* Monitor system activity and logs.

**How to Use the Application**

1. **Log In**:
   * Use the credentials created during the setup process or provided by another administrator.
   * Navigate to the admin panel via the /admin URL.
2. **Manage Users**:
   * Access the "Users" section in the admin panel.
   * Add new users by filling in their details and assigning appropriate roles (e.g., Project Manager, Developer, Client).
   * Edit or deactivate existing user accounts as needed.
3. **Configure Permissions**:
   * Assign specific permissions to roles or individual users to control access to various features.
   * Use the "Groups" feature to manage permissions for multiple users at once.
4. **Monitor System Activity**:
   * Review system logs and activity reports to track user actions and ensure smooth operation.
   * Address any system errors or issues.
5. **Manage Projects** (Optional):
   * If required, create and assign projects to Project Managers or Developers.

**2. Project Manager Role**

The Project Manager oversees project creation, task assignment, and progress tracking.

**Key Responsibilities**

* Create and manage projects.
* Assign tasks to developers.
* Track project progress and ensure deadlines are met.

**How to Use the Application**

1. **Log In**:
   * Use the credentials provided by the Administrator.
2. **Create a New Project**:
   * Navigate to the "Projects" section.
   * Click on "Create New Project" and fill in the required details, such as project name, description, and deadlines.
3. **Add Team Members**:
   * Assign developers to the project by selecting users from the list of available team members.
   * Specify roles and responsibilities for each team member.
4. **Create and Assign Tasks**:
   * Within a project, navigate to the "Tasks" section.
   * Click "Add Task" to create a new task, providing details like task title, description, due date, and priority.
   * Assign the task to a specific developer.
5. **Monitor Progress**:
   * Use the project dashboard to track the status of tasks and overall project progress.
   * Review reports and timelines to ensure deadlines are being met.
6. **Communicate with Team Members**:
   * Use built-in messaging or comment features to communicate with developers and clients about project updates or issues.

**3. Developer Role**

Developers are responsible for working on assigned tasks and updating their progress.

**Key Responsibilities**

* View assigned tasks and deadlines.
* Update task status and provide progress details.
* Communicate with the Project Manager and team.

**How to Use the Application**

1. **Log In**:
   * Use the credentials provided by the Administrator or Project Manager.
2. **View Assigned Tasks**:
   * Navigate to the "My Tasks" section to see a list of tasks assigned to you.
   * Each task will display details such as the title, description, deadline, and priority.
3. **Update Task Status**:
   * Open a task and update its status (e.g., "In Progress," "Completed").
   * Add comments or upload files if additional information is required.
4. **Track Deadlines**:
   * Use the task timeline or calendar view to keep track of upcoming deadlines.
5. **Collaborate**:
   * Communicate with the Project Manager or other team members using the built-in messaging or comment features.
   * Raise any issues or roadblocks that might impact task completion.

**4. Client/User Role**

Clients or end-users can submit issues, track their status, and provide feedback on resolved issues.

**Key Responsibilities**

* Submit new issues or bug reports.
* Track the status of submitted issues.
* Provide feedback on resolved issues.

**How to Use the Application**

1. **Log In**:
   * Use the credentials provided by the Administrator or Project Manager.
2. **Submit a New Issue**:
   * Navigate to the "Submit Issue" section.
   * Fill out the issue submission form, providing details such as the issue title, description, priority, and any relevant attachments.
3. **Track Issue Status**:
   * Go to the "My Issues" section to view a list of issues you have submitted.
   * Each issue will display its current status (e.g., "Open," "In Progress," "Resolved") and any comments from the development team.
4. **Provide Feedback**:
   * Once an issue is marked as resolved, review the resolution and provide feedback.
   * Reopen the issue if it has not been resolved to your satisfaction.
5. **Communicate with the Team**:
   * Use the comment section within an issue to communicate directly with the team handling the issue.

**5. General Features Available to All Roles**

The following features are accessible to all user roles, depending on their permissions:

* **Dashboard**:
  + The dashboard provides an overview of tasks, projects, or issues relevant to the logged-in user.
  + Users can view summaries, deadlines, and notifications for their role-specific responsibilities.
* **Notifications**:
  + The application sends notifications for important events, such as task assignments, status updates, or issue resolutions.
  + Notifications can be accessed via the notification panel or email, depending on the configuration.
* **Search and Filters**:
  + Use the search bar or filters to quickly locate projects, tasks, or issues based on criteria like status, priority, or deadline.
* **Profile Management**:
  + Users can update their personal information, such as name, email, and password, in the profile section.

**6. Best Practices for Using the Application**

* **Regular Updates**:
  + Developers and Project Managers should update task statuses regularly to ensure accurate progress tracking.
  + Clients should promptly review resolved issues and provide feedback.
* **Clear Communication**:
  + Use the built-in messaging and comment features to maintain clear and consistent communication among team members.
* **Monitor Deadlines**:
  + Use the dashboard and calendar views to stay informed about upcoming deadlines and prioritize tasks accordingly.
* **Role-Specific Actions**:
  + Perform actions only within your role’s scope to avoid conflicts or unauthorized changes.

This user manual ensures that all users can effectively navigate and utilize the **Software Project Issue Tracker** application according to their roles. By following these guidelines, users can streamline their workflows and contribute to the success of their projects or issue resolutions.

**API Documentation**

**Overview**

This API provides endpoints to manage and view records for projects, tasks, and users. It is designed for efficient data retrieval and management in a project issue tracker system. Authentication is required for all endpoints, and certain endpoints are restricted to admin users.

**Endpoints**

**Projects**

1. **GET /api/projects/**
   * **Description**: Retrieve a list of all projects.
   * **Authentication**: Required.
   * **Response**:
     + **200 OK**: Returns a list of all projects with details such as ID, name, description, creation date, and last updated date.
   * **Error Codes**:
     + **401 Unauthorized**: If the user is not authenticated.
2. **GET /api/projects/<project\_id>/**
   * **Description**: Retrieve details of a specific project by its ID.
   * **Authentication**: Required.
   * **Response**:
     + **200 OK**: Returns the project details.
     + **404 Not Found**: If the project with the specified ID does not exist.
   * **Error Codes**:
     + **401 Unauthorized**: If the user is not authenticated.

**Tasks**

1. **GET /api/tasks/**
   * **Description**: Retrieve a list of all tasks.
   * **Authentication**: Required.
   * **Response**:
     + **200 OK**: Returns a list of all tasks with details such as ID, title, description, status, associated project ID, creation date, and last updated date.
   * **Error Codes**:
     + **401 Unauthorized**: If the user is not authenticated.
2. **GET /api/tasks/<task\_id>/**
   * **Description**: Retrieve details of a specific task by its ID.
   * **Authentication**: Required.
   * **Response**:
     + **200 OK**: Returns the task details.
     + **404 Not Found**: If the task with the specified ID does not exist.
   * **Error Codes**:
     + **401 Unauthorized**: If the user is not authenticated.

**Users**

1. **GET /api/users/**
   * **Description**: Retrieve a list of all users (Admin only).
   * **Authentication**: Admin users only.
   * **Response**:
     + **200 OK**: Returns a list of all users with details such as ID, username, email, first name, last name, staff status, and date joined.
   * **Error Codes**:
     + **401 Unauthorized**: If the user is not authenticated.
     + **403 Forbidden**: If the user is not an admin.
2. **GET /api/users/<user\_id>/**
   * **Description**: Retrieve details of a specific user by their ID (Admin only).
   * **Authentication**: Admin users only.
   * **Response**:
     + **200 OK**: Returns the user details.
     + **404 Not Found**: If the user with the specified ID does not exist.
   * **Error Codes**:
     + **401 Unauthorized**: If the user is not authenticated.
     + **403 Forbidden**: If the user is not an admin.

**Authentication**

* All endpoints require authentication via token-based authentication or session-based authentication.
* Admin-only endpoints require the user to have staff privileges.

**Error Codes**

* **401 Unauthorized**: Occurs when the user is not authenticated.
* **403 Forbidden**: Occurs when the user does not have the necessary permissions for the endpoint.
* **404 Not Found**: Occurs when the requested resource (e.g., project, task, or user) does not exist.

**Response Format**

All responses are returned in JSON format. The structure of the responses depends on the endpoint and the resource being accessed.

**Testing the API**

1. **Swagger UI**:
   * Navigate to /swagger/ to test the API interactively.
2. **ReDoc**:
   * Navigate to /redoc/ for a clean and structured view of the API documentation.
3. **Postman**:
   * Export the Swagger JSON schema from /swagger.json and import it into Postman for testing.