Leave Management System Administrator Manual

Introduction

This Administrator Manual provides instructions for system administrators to manage the Leave Management System (LMS), a Django-based application for handling employee leave requests and balances.

System Overview

The LMS allows:

- Employees to log in, view leave balances, submit leave requests, and view request statuses.
- Managers to review and approve/reject leave requests.
- Administrators to manage employee accounts, leave balances, and system configurations.

Prerequisites

- Access to the Django Admin Panel (http://127.1.0.1:8080/admin/).
- Superuser credentials for the Django Admin Panel.
- Basic knowledge of Django and database management.

Administrator Responsibilities

- Create and manage employee accounts.
- Initialize and update leave balances.
- Monitor system performance and resolve errors.
- Ensure data integrity and security.

Accessing the Django Admin Panel

1. Navigate to the admin URL (http://127.1.0.1:8080/admin/).
2. Log in with your superuser credentials (username and password).

Managing Employee Accounts

1. **Access the Employee Model**:
    - In the Django Admin Panel, locate the **Employee** model under the application name (e.g., lms_app).
2. **Add a New Employee**:

- Click **Add Employee**.
- Fill in:
  - **Employee ID**: Unique integer (primary key).
  - **Full Name**: Employee's full name (max 100 characters).
  - **Position**: Select Officer, Senior, Principal, or Manager.
  - **Employment Date**: Optional date of employment.
  - **Username**: Set to Employee ID (must be unique).
  - **Password**: Set a secure password (use "Set Password" form).
- Save the employee.

3. **Edit an Employee**:
   - Select an employee from the list.
   - Update fields as needed and save.
4. **Delete an Employee**:
   - Select an employee and choose "Delete" from the action dropdown.
   - Note: Deleting an employee will cascade to related leave balances and requests.

Managing Leave Balances

1. **Access the Leave Balance Model**:
   - Locate the **LeaveBalance** model in the Django Admin Panel.
2. **Add a Leave Balance**:
   - Click **Add Leave Balance**.
   - Select the **Employee** from the dropdown.
   - Set **Annual Leave Balance** (default: 20 days).
   - Set **Sick Leave Balance** (default: 15 days).
   - Save the balance.
3. **Edit a Leave Balance**:
   - Select a balance from the list.
   - Update fields and save.

- Ensure balances reflect approved leave deductions.

4. **Delete a Leave Balance**:
    - Select a balance and delete it (only if necessary, as it's linked to an employee).

Managing Leave Requests

1. **Access the Leave Request Model**:
    - Locate the **LeaveRequest** model in the Django Admin Panel.
2. **View Requests**:
    - View all leave requests, including employee, leave type, dates, status, and comments.
3. **Edit a Request**:
    - Select a request to update status or comments manually (e.g., for administrative corrections).
    - Ensure balance updates if status changes to Approved.
4. **Delete a Request**:
    - Delete requests if invalid or erroneous (use cautiously).

Monitoring and Troubleshooting

1. **Check Login Issues**:
    - Verify employee usernames and passwords in the Employee model.
    - Reset passwords if employees cannot log in.
2. **Leave Balance Errors**:
    - If an employee sees "Leave balance not found", ensure a LeaveBalance record exists for them.
    - Create a new LeaveBalance record if missing.
3. **Form Submission Issues**:
    - Check form validation in forms.py (e.g., rejection comment requirement).
    - Review server logs for errors (DEBUG=True in development).
4. **Database Integrity**:
    - Ensure no duplicate Employee IDs or usernames.

  - Verify foreign key relationships (e.g., LeaveRequest.employee, LeaveBalance.employee).

System Maintenance

- **Backups**:
  - Regularly back up the database to prevent data loss.
  - Use Django's dumpdata command or database-specific tools.
- **Updates**:
  - Keep Django and dependencies updated (e.g., pip install --upgrade django).
  - Test updates in a development environment first.
- **Security**:
  - Use strong passwords for superuser accounts.
  - Enable HTTPS on the production server.
  - Restrict admin panel access to authorized IPs.

Contact

For technical support, contact the development team or refer to Django documentation.