Disease Prediction System

Pneumonia Detector

Project Documentation

Generated on: November 09, 2025

Abstract

This document provides comprehensive documentation for the Disease Prediction System, specifically designed for pneumonia detection using chest X-ray images. The system is built using Django web framework and employs a deep learning model (TensorFlow/Keras) to classify chest X-ray images as either "Normal" or "Pneumonia". The application features user authentication, image upload functionality, real-time prediction capabilities, and a comprehensive dashboard for tracking prediction history and statistics. This documentation covers all modules, their functionality, architecture, and implementation details.

Table of Contents

1. Introduction

The Disease Prediction System is a web-based application that uses artificial intelligence to detect pneumonia from chest X-ray images. The system is designed to assist healthcare professionals and researchers in analyzing medical images efficiently. **Key Features:**
• User authentication and authorization
• Image upload and preprocessing
• Real-time pneumonia prediction using deep learning
• Prediction history and statistics dashboard
• User-friendly web interface
• Secure file handling and storage

1.1 Technology Stack

Backend Framework: Django 4.2.25
Machine Learning: TensorFlow 2.15.0, Keras 2.15.0
Database: SQLite3
Image Processing: Pillow 11.3.0, NumPy 1.26.4
Frontend: HTML5, CSS3, JavaScript
Server: Django Development Server (WSGI/ASGI)

2. Project Structure

The project follows Django's standard project structure with the following key directories:

```
disease_prediction/
├── disease_prediction/          # Core project configuration
│   ├── __init__.py
│   ├── settings.py              # Django settings
│   ├── urls.py                  # Main URL configuration
│   ├── wsgi.py                  # WSGI configuration
│   └── asgi.py                  # ASGI configuration
├── pneumonia_predictor/         # Main application
│   ├── __init__.py
│   ├── models.py                # Database models
│   ├── views.py                 # View functions
│   ├── forms.py                 # Form definitions
│   ├── admin.py                 # Admin configuration
│   ├── apps.py                  # App configuration
│   ├── templates/               # HTML templates
│   │   ├── base.html
│   │   ├── upload.html
│   │   ├── dashboard.html
│   │   └── registration/
│   │       ├── login.html
│   │       └── signup.html
│   └── models/                  # ML model storage
│       └── pneumonia_model.h5
├── media/                       # Uploaded media files
├── db.sqlite3                   # SQLite database
├── manage.py                    # Django management script
└── requirements.txt             # Python dependencies
```

3. Core Module (disease_prediction)

The core module contains the main Django project configuration.

3.1 settings.py

The settings.py file contains all Django project configuration including:
• **INSTALLED_APPS:** Lists all installed Django applications including 'pneumonia_predictor'
• **MIDDLEWARE:** Configures security, session, authentication, and CSRF middleware
• **DATABASES:** SQLite3 database configuration
• **STATIC_URL and MEDIA_URL:** Configuration for static and media files
• **AUTH_PASSWORD_VALIDATORS:** Password validation rules for user authentication
• **LOGIN_URL, LOGIN_REDIRECT_URL, LOGOUT_REDIRECT_URL:** Authentication URL configuration
• **SECRET_KEY:** Secret key for cryptographic signing (should be kept secure in production)
• **DEBUG:** Debug mode (set to False in production)
• **ALLOWED_HOSTS:** List of allowed hostnames

3.2 urls.py

The main URL configuration file routes URLs to their respective views:
• **'/admin/':** Django admin interface
• **'/':** Main upload and prediction page (upload_and_predict view)
• **'/dashboard/':** Dashboard view showing prediction statistics
• **'/accounts/signup/':** User registration page
• **'/accounts/':** Includes Django's built-in authentication URLs (login, logout, password reset)
• **Media files:** Serves uploaded media files during development

3.3 wsgi.py and asgi.py

wsgi.py: Web Server Gateway Interface configuration for deployment with WSGI servers (e.g., Gunicorn, uWSGI)
asgi.py: Asynchronous Server Gateway Interface configuration for deployment with ASGI servers (e.g., Daphne, Uvicorn) supporting WebSockets and async features

4. Pneumonia Predictor Module

The pneumonia_predictor module is the main application containing the core functionality.

4.1 views.py

The views.py file contains all view functions that handle HTTP requests:

4.1.1 predict_pneumonia()

Function: predict_pneumonia(img_path)
Purpose: Predicts whether a chest X-ray shows pneumonia or is normal
Parameters: img_path (str) - Path to the image file
Returns: Dictionary with 'label' ('Normal' or 'Pneumonia') and 'probability' (float)
Process:
1. Loads and preprocesses the image (resize to 224x224, normalize to [0,1])
2. Uses the loaded TensorFlow/Keras model to make predictions
3. Returns prediction label and confidence score
Model: Loaded once at application startup from pneumonia_model.h5

4.1.2 upload_and_predict()

Function: upload_and_predict(request) **Decorator:** @login_required(login_url='login')
Purpose: Handles image upload and prediction requests
Process:
1. Validates uploaded image using UploadImageForm
2. Saves uploaded image to media directory
3. Calls predict_pneumonia() to get prediction
4. Saves prediction results to database (Prediction model)
5. Renders upload.html template with results
Template: upload.html

4.1.3 dashboard()

Function: dashboard(request) **Decorator:** @login_required(login_url='login')
Purpose: Displays prediction statistics and history
Data Provided:
• Total number of predictions
• Number of pneumonia cases detected
• Number of normal cases

• Recent predictions for the current user (last 20)
Template: dashboard.html

4.1.4 signup()

Function: signup(request)
Purpose: Handles user registration
Process:
1. Uses Django's UserCreationForm for user registration
2. Validates form data
3. Creates new user account
4. Logs in the user automatically after registration
5. Redirects to upload_and_predict page
Template: registration/signup.html

4.2 models.py

The models.py file defines the database models:

4.2.1 Prediction Model

Model: Prediction
Fields:
• user (ForeignKey): Links prediction to a user account
• image_name (CharField): Name of the uploaded image file
• label (CharField): Prediction result ('Normal' or 'Pneumonia')
• probability (FloatField): Confidence score from the model
• created_at (DateTimeField): Timestamp of when prediction was made (auto-generated)
Meta Options:
• ordering: ['-created_at'] - Orders predictions by most recent first
Methods:
• image_url(): Returns the URL path to the uploaded image

4.3 forms.py

The forms.py file defines form classes for user input:

4.3.1 UploadImageForm

Form: UploadImageForm
Fields:
• image (ImageField): Field for uploading chest X-ray images
Validation: Django automatically validates that uploaded file is a valid image format
Usage: Used in upload_and_predict view for image upload functionality

4.4 admin.py

The admin.py file is used to register models with Django's admin interface. Currently, no models are registered, but the Prediction model can be registered for administrative management of prediction records.

5. Authentication Module

The application uses Django's built-in authentication system (django.contrib.auth) for user management and authentication.

5.1 Authentication Features

User Registration:
• Custom signup view using UserCreationForm
• Automatic login after successful registration
• Password validation (minimum 8 characters, common password checks)
User Login:
• Django's built-in login view
• Session-based authentication
• Redirects to upload page after login
User Logout:
• Django's built-in logout view
• Redirects to login page after logout
Password Management:
• Password reset functionality (via Django's built-in views)
• Password change functionality
Access Control:
• @login_required decorator protects views
• Unauthenticated users are redirected to login page

5.2 Authentication URLs

The following authentication URLs are available:
• /accounts/login/ - User login page
• /accounts/logout/ - User logout
• /accounts/signup/ - User registration page
• /accounts/password_reset/ - Password reset request
• /accounts/password_reset/done/ - Password reset confirmation
• /accounts/reset/// - Password reset form
• /accounts/reset/done/ - Password reset complete

5.3 User Model

The application uses Django's default User model which includes:
• username - Unique username for each user
• password - Hashed password stored securely
• email - User email address (optional)
• first_name, last_name - User's name (optional)
• is_active - Whether the user account is active
• is_staff - Whether the user can access admin site
• is_superuser - Whether the user has all permissions
• date_joined - Account creation timestamp

6. Database Models

The application uses SQLite3 as the database (suitable for development and small deployments). For production, consider using PostgreSQL or MySQL.

6.1 Database Schema

Prediction Table:
• id (Primary Key, Auto-increment)
• user_id (Foreign Key to User table)
• image_name (VARCHAR 255)
• label (VARCHAR 32)
• probability (FLOAT)
• created_at (DATETIME)
Relationships:
• One User can have many Predictions (One-to-Many)
• Predictions are cascade deleted when User is deleted

6.2 Database Migrations

Database migrations are stored in pneumonia_predictor/migrations/ directory.
Initial Migration: 0001_initial.py creates the Prediction table
Running Migrations: python manage.py migrate
Creating Migrations: python manage.py makemigrations

7. URL Routing and Views

The application uses Django's URL routing system to map URLs to view functions.

7.1 URL Patterns

URL Pattern	View Function	Name	Requires Auth
/admin/	admin.site.urls	admin	Yes (Staff)
/	upload_and_predict	upload_and_predict	Yes
/dashboard/	dashboard	dashboard	Yes
/accounts/signup/	signup	signup	No
/accounts/login/	django.contrib.auth.views.LoginView	login	No
/accounts/logout/	django.contrib.auth.views.LogoutView	logout	No
/media/<path>	serve media files	N/A	No

8. Templates and User Interface

The application uses Django's template system with HTML templates located in pneumonia_predictor/templates/ directory.

8.1 Base Template (base.html)

Purpose: Base template that all other templates extend
Features:
• Responsive navigation bar with user authentication status
• Links to Predict, Dashboard, Login, Logout, Sign Up
• User badge showing current logged-in user
• Modern CSS styling with gradient themes
• Mobile-responsive design
• Block system for title, content, extra_head, and extra_scripts

8.2 Upload Template (upload.html)

Purpose: Main page for uploading chest X-ray images and viewing predictions
Features:
• Drag-and-drop file upload zone
• Image preview before submission
• Real-time prediction results display
• Confidence meter visualization
• Prediction label (Normal/Pneumonia) with color coding
• Medical disclaimer notice
• Responsive two-column layout (upload form + results panel)

8.3 Dashboard Template (dashboard.html)

Purpose: Display prediction statistics and history
Features:
• Statistics cards showing total predictions, pneumonia cases, normal cases
• Recent predictions table with image thumbnails
• Prediction labels with color-coded badges
• Confidence scores and timestamps
• User-specific prediction history
• Information sidebar with usage tips
• Responsive grid layout

8.4 Authentication Templates

Login Template (registration/login.html):
• User login form with username and password fields
• Error message display for invalid credentials
• Link to signup page
• Modern card-based design
Signup Template (registration/signup.html):
• User registration form
• Password validation requirements
• Form error display
• Link to login page
• Automatic login after successful registration

9. Installation and Setup

Follow these steps to set up and run the Disease Prediction System:

9.1 Prerequisites

Required Software:
• Python 3.8 or higher
• pip (Python package installer)
• Virtual environment (recommended)
• TensorFlow-compatible system (CPU or GPU)

9.2 Installation Steps

Step 1: Clone or Download Project
Download or clone the project to your local machine.
Step 2: Create Virtual Environment
python -m venv venv
Step 3: Activate Virtual Environment
Windows: venv\Scripts\activate
Linux/Mac: source venv/bin/activate
Step 4: Install Dependencies
pip install -r requirements.txt
Step 5: Run Migrations
python manage.py migrate
Step 6: Create Superuser (Optional)
python manage.py createsuperuser
This allows you to access the Django admin interface.
Step 7: Ensure Model File Exists
Make sure the pneumonia_model.h5 file is located in pneumonia_predictor/models/ directory.
The model file is required for predictions to work.

9.3 Running the Application

Start Development Server:
python manage.py runserver
The application will be available at http://127.0.0.1:8000/
Access Admin Interface:
Navigate to http://127.0.0.1:8000/admin/ and login with superuser credentials

Stop Server:
Press Ctrl+C in the terminal to stop the development server
Note: The development server is not suitable for production. For production deployment, use a proper WSGI/ASGI server like Gunicorn or uWSGI with a reverse proxy like Nginx.

10. Usage Instructions

This section provides step-by-step instructions for using the Disease Prediction System.

10.1 User Registration

Step 1: Access Registration Page
Navigate to http://127.0.0.1:8000/accounts/signup/ or click the "Sign Up" link in the navigation bar.
Step 2: Fill Registration Form
• Enter a unique username
• Enter a secure password (minimum 8 characters)
• Confirm the password
Step 3: Submit Registration
Click the "Sign Up" button. Upon successful registration, you will be automatically logged in and redirected to the upload page.

10.2 User Login

Step 1: Access Login Page
Navigate to http://127.0.0.1:8000/accounts/login/ or click the "Login" link in the navigation bar.
Step 2: Enter Credentials
• Enter your username
• Enter your password
Step 3: Submit Login
Click the "Login" button. Upon successful login, you will be redirected to the upload page.

10.3 Making Predictions

Step 1: Access Upload Page
Navigate to http://127.0.0.1:8000/ or click "Predict" in the navigation bar (requires login).
Step 2: Upload Chest X-ray Image
• Click the upload area or drag and drop a chest X-ray image file
• Supported formats: JPEG, PNG, GIF, BMP
• Recommended image size: 224x224 pixels or larger (will be resized automatically)
Step 3: View Prediction Results
After uploading, the system will:
• Display the uploaded image
• Show prediction label (Normal or Pneumonia)

• Display confidence score (probability)
• Show a visual confidence meter

Step 4: Review Results

The prediction result is automatically saved to your account and can be viewed in the Dashboard.

Important: This tool is for educational/research purposes only. Always consult with healthcare professionals for medical diagnoses.

10.4 Viewing Dashboard

Step 1: Access Dashboard

Navigate to http://127.0.0.1:8000/dashboard/ or click "Dashboard" in the navigation bar (requires login).

Step 2: View Statistics

The dashboard displays:

• Total number of predictions made
• Number of pneumonia cases detected
• Number of normal cases detected

Step 3: View Prediction History

• Scroll through your recent predictions (last 20)
• View image thumbnails, prediction labels, confidence scores, and timestamps
• Predictions are sorted by most recent first

10.5 User Logout

Step 1: Click Logout

Click the "Logout" link in the navigation bar.

Step 2: Confirmation

You will be logged out and redirected to the login page. You must log in again to access protected features.

11. API Endpoints

The application provides web-based endpoints (URLs) for accessing different features. While this is not a REST API, the URL endpoints serve as the interface to the application.

11.1 Public Endpoints

Login Page: GET /accounts/login/
• Purpose: Display user login form
• Method: GET (display form), POST (submit credentials)
• Authentication: Not required
Registration Page: GET /accounts/signup/
• Purpose: Display user registration form
• Method: GET (display form), POST (submit registration)
• Authentication: Not required
Password Reset: GET /accounts/password_reset/
• Purpose: Request password reset
• Method: GET, POST
• Authentication: Not required

11.2 Protected Endpoints

Upload and Predict: GET, POST /
• Purpose: Upload chest X-ray image and get prediction
• Method: GET (display form), POST (upload image and predict)
• Authentication: Required (login_required decorator)
• Parameters: image (file upload in POST request)
• Returns: HTML page with prediction results
Dashboard: GET /dashboard/
• Purpose: Display prediction statistics and history
• Method: GET
• Authentication: Required (login_required decorator)
• Returns: HTML page with statistics and recent predictions
Logout: POST /accounts/logout/
• Purpose: Log out current user
• Method: POST
• Authentication: Required
• Returns: Redirects to login page
Admin Interface: GET /admin/
• Purpose: Django admin interface for managing data
• Method: GET, POST

• Authentication: Required (staff user)

11.3 Media Files

Media Files: GET /media/
• Purpose: Serve uploaded images
• Method: GET
• Authentication: Not required (in development)
• Note: In production, configure proper media file serving through web server (Nginx, Apache)

12. Dependencies

The following are the main Python packages required for this project. A complete list is available in requirements.txt. Install all dependencies using: pip install -r requirements.txt

12.1 Core Dependencies

Django 4.2.25: Web framework for building the application
TensorFlow 2.15.0: Machine learning framework for loading and running the pneumonia prediction model
Keras 2.15.0: High-level neural networks API (included with TensorFlow)
NumPy 1.26.4: Numerical computing library for array operations
Pillow 11.3.0: Python Imaging Library for image processing and manipulation
h5py 3.14.0: Python interface to the HDF5 binary data format (required for loading .h5 model files)

12.2 Supporting Dependencies

asgiref 3.10.0: ASGI specification and utilities (Django dependency)
sqlparse 0.5.3: SQL parser for Django database operations
tzdata 2025.2: Timezone database for datetime operations
MarkupSafe 3.0.3: Safe string handling for templates
Werkzeug 3.1.3: WSGI utilities (used by Django development server)

12.3 TensorFlow Dependencies

TensorFlow requires several additional packages for proper functioning:
protobuf 4.25.8: Protocol buffers for serializing structured data
grpcio 1.75.1: gRPC library for communication
flatbuffers 25.9.23: Serialization library for efficient data storage
tensorboard 2.15.2: Visualization toolkit for TensorFlow (optional, for model training)
gast 0.4.0: Generic AST (Abstract Syntax Tree) for Python
opt_einsum 3.4.0: Optimized einsum function for tensor operations
ml-dtypes 0.2.0: Machine learning data types
termcolor 3.1.0: ANSI color formatting for terminal output

12.4 Installation Notes

System Requirements:
• Python 3.8 or higher
• TensorFlow supports both CPU and GPU. For GPU support, install TensorFlow-GPU and CUDA toolkit
• Minimum 4GB RAM recommended (8GB+ for better performance)
• Disk space: At least 2GB for dependencies and model files
Installation Tips:
• Use a virtual environment to avoid conflicts with other projects
• On Windows, you may need to install Visual C++ Redistributable for TensorFlow
• For production, consider using a lighter deployment option or containerization (Docker)
• Regularly update dependencies for security patches: pip install --upgrade -r requirements.txt

Conclusion

This documentation provides a comprehensive overview of the Disease Prediction System (Pneumonia Detector). The system leverages deep learning and web technologies to provide an accessible platform for chest X-ray image analysis.

Key Takeaways:
• The system uses a pre-trained TensorFlow/Keras model for pneumonia detection
• Django framework provides a robust and secure web interface
• User authentication ensures data privacy and user-specific prediction history
• The dashboard provides valuable insights into prediction statistics

Future Enhancements:
• REST API for programmatic access
• Support for multiple disease detection
• Model retraining capabilities
• Advanced visualization and analytics
• Integration with healthcare systems
• Mobile application support

Important Disclaimer:
This system is intended for educational and research purposes only. It should not be used as a substitute for professional medical advice, diagnosis, or treatment. Always consult with qualified healthcare professionals for medical decisions.