In [1]:
```python
## Preparation

# Import the necessary libraries
import torch
import torch.nn as nn
from sklearn.model_selection import train_test_split
from torchvision import models, datasets, transforms
from torch.utils.data import DataLoader, Subset
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classificatio
import matplotlib.pyplot as plt
import numpy as np
import os
import optuna
import time

from activation import Swish, ResidualBlock

## Global Var

dataset_path = os.getcwd() + '/dataset/classified'

class_labels = ["Normal", "Cataract"]
```

In [2]:
```python
print(f"Is using CUDA? {torch.cuda.is_available()}")  # Should return True if CUDA
print(torch.version.cuda)  # Check the CUDA version PyTorch is using
print(torch.cuda.current_device()) # Check CUDA device used
```

```
Is using CUDA? True
12.6
0
```

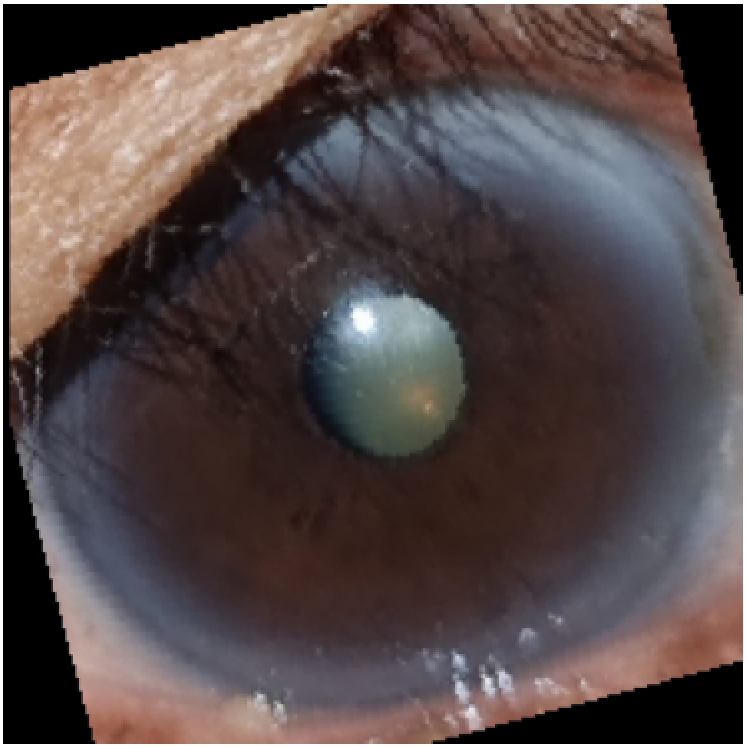
In [3]:
```python
# Augmentation
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomRotation(15),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomCrop(224, padding=4),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize
])

# Load Dataset
ds = datasets.ImageFolder(root=dataset_path, transform=transform)

indices = list(range(len(ds)))
# labels = [ds.targets[i] for i in indices]

# Split into train and test dataset
train_indices, test_indices = train_test_split(indices, test_size=0.3, random_state

train_ds = Subset(ds, train_indices)
test_ds = Subset(ds, test_indices)

train_loader = DataLoader(train_ds, batch_size=128, shuffle=True, pin_memory=True,
```

```
test_loader = DataLoader(test_ds, batch_size=128, shuffle=False, pin_memory=True, n

total_samples  = len(train_ds) + len(test_ds)

print(f"Train size: {(len(train_ds) / total_samples) * 100:.2f}%, Test size: {(len(
print(f"Total samples: {total_samples}, Train size: {len(train_ds)}, Test size: {le
```

```
Train size: 69.97%, Test size: 30.03%
Total samples: 1159, Train size: 811, Test size: 348
```

In [4]:
```
def denormalize(tensor, mean=None, std=None):
    if std is None:
        std = [0.5, 0.5, 0.5]
    if mean is None:
        mean = [0.5, 0.5, 0.5]
    mean = torch.tensor(mean).view(3, 1, 1)
    std = torch.tensor(std).view(3, 1, 1)
    return tensor * std + mean  # Reverse normalization

# Get a batch of images
dataiter = iter(train_loader)
images, labels = next(dataiter)

# Select one image
img = images[0]
label = labels[0].item()

# Denormalize image
img = denormalize(img)

# Convert from Tensor (C, H, W) to NumPy (H, W, C)
img = np.transpose(img.numpy(), (1, 2, 0))

# Plot the image
plt.imshow(img)
plt.title(f"Label: {class_labels[label]}")  # Display label
plt.axis("off")
plt.show()
```

Label: Cataract



In [5]:
```
# Optuna Hyperparameter

best_trial = None
best_model = None

torch.backends.cudnn.benchmark = True

device_name = "cuda" if torch.cuda.is_available() else "cpu"
use_amp = device_name == "cuda"


def objective(trial: optuna.Trial) -> float:
    device = torch.device(device_name)

    # Define hyperparameters
    lr = trial.suggest_float("lr", 1e-3, 1e-1, log=True)
    dropout_rate = trial.suggest_float("dropout", 0.3, 0.5)
    optimizer_name = trial.suggest_categorical("optimizer", ["AdamW", "SGD"])
    num_epochs = trial.suggest_int("num_epochs", 5, 15)

    model = models.efficientnet_b0(progress=True, weights=models.EfficientNet_B0_We

    for param in model.features[:-2].parameters():
        param.requires_grad = False

    number_of_features = model.classifier[1].in_features
    model.classifier = nn.Sequential(
        nn.Linear(number_of_features, 256),
        nn.BatchNorm1d(256),
        Swish(),
```

```
        ResidualBlock(256, 256),
        nn.Dropout(dropout_rate),

        ResidualBlock(256, 128),
        nn.Dropout(dropout_rate),

        ResidualBlock(128, 128),
        nn.Dropout(dropout_rate),

        ResidualBlock(128, 64),
        nn.Dropout(dropout_rate),

        nn.Linear(64, 1)
    ).to(device)

    model.to(device)

    # Define loss and optimizer
    criterion = nn.BCEWithLogitsLoss()

    optimizer = torch.optim.AdamW(model.parameters(), lr=lr, weight_decay=1e-4)
    if optimizer_name == "SGD":
        optimizer = torch.optim.SGD(model.parameters(), lr=lr, momentum=0.9)

    scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=10)

    train_losses = []
    train_accuracies = []

    scaler = torch.amp.GradScaler(device=device_name)

    # Training Loop
    for epoch in range(num_epochs):

        model.train()
        total_loss, correct, total = 0, 0, 0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.float().to(device).unsqueeze

            optimizer.zero_grad()

            with torch.amp.autocast("cuda", enabled=use_amp):
                outputs = model(images)
                loss = criterion(outputs, labels)

            scaler.scale(loss).backward()
            scaler.step(optimizer)
            scaler.update()

            total_loss += loss.item()
            preds = (torch.sigmoid(outputs) > 0.5).float()
            correct += (preds == labels).sum().item()
            total += labels.size(0)

        scheduler.step()
```

```
        epoch_loss = total_loss / len(train_loader)
        epoch_acc = correct / total * 100

        train_losses.append(epoch_loss)
        train_accuracies.append(epoch_acc)

        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Accuracy: {

        trial.set_user_attr("train_losses", train_losses)
        trial.set_user_attr("train_accuracies", train_accuracies)

        # Pruning: Stop bad trials early
        trial.report(epoch_acc, epoch)
        if trial.should_prune():
            raise optuna.exceptions.TrialPruned()

    # Evaluate Model
    model.eval()
    correct, total = 0, 0
    trial_preds = []
    trial_labels = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.float().to(device).unsqueeze
            outputs = model(images)
            preds = (torch.sigmoid(outputs) > 0.5).float()

            correct += (preds == labels).sum().item()
            total += labels.size(0)

            trial_preds.extend(preds.cpu().numpy())
            trial_labels.extend(labels.cpu().numpy())

    test_acc = correct / total * 100
    print(f"🎯 Test Accuracy: {test_acc:.2f}%")

    # **Save best model globally**
    global best_model, best_trial

    try:
        best_trial = trial.study.best_trial
        best_value = trial.study.best_value
    except ValueError:
        best_value = float('-inf')

    if best_model is None or (best_trial is not None and test_acc > best_value):
        best_model = model.state_dict()

    best_acc = trial.study.user_attrs.get("best_accuracy", 0)

    if test_acc > best_acc:
        study.set_user_attr("best_accuracy", test_acc)

        trial_preds = np.array(trial_preds).flatten().tolist()
```

```python
        trial_labels = np.array(trial_labels).flatten().tolist()

        study.set_user_attr("best_preds", trial_preds)
        study.set_user_attr("best_labels", trial_labels)

    return test_acc
```

In [6]:
```python
study = optuna.create_study(
    direction="maximize",
    study_name=f"hyperparam cataract classifier_{int(time.time())}",
    pruner=optuna.pruners.MedianPruner(),
    storage="sqlite:///optuna.db",
    load_if_exists=True
)

study.optimize(
    objective,
    n_trials=50,
    n_jobs=4,
    show_progress_bar=True
)

acc = study.best_value

print("\nBest Hyperparameters:", study.best_params)
print("\nAccuracy:", acc)
```

```
[I 2025-03-13 20:14:22,334] A new study created in RDB with name: hyperparam catarac
t classifier_1741871661
  0%|          | 0/50 [00:00<?, ?it/s]
```

```
Epoch [1/10], Loss: 0.6744, Accuracy: 58.08%
Epoch [1/11], Loss: nan, Accuracy: 50.06%
Epoch [1/8], Loss: 0.6102, Accuracy: 61.90%
Epoch [1/13], Loss: 0.6377, Accuracy: 61.78%
Epoch [2/13], Loss: 0.4640, Accuracy: 76.70%
Epoch [2/10], Loss: 0.6631, Accuracy: 60.05%
Epoch [2/8], Loss: 0.3783, Accuracy: 87.42%
Epoch [2/11], Loss: nan, Accuracy: 49.45%
Epoch [3/10], Loss: 0.6461, Accuracy: 64.98%
Epoch [3/13], Loss: 0.3024, Accuracy: 87.55%
Epoch [3/8], Loss: 0.2599, Accuracy: 93.59%
Epoch [3/11], Loss: nan, Accuracy: 54.50%
Epoch [4/13], Loss: 0.2204, Accuracy: 91.74%
Epoch [4/10], Loss: 0.6309, Accuracy: 66.71%
Epoch [4/8], Loss: 0.2100, Accuracy: 94.33%
Epoch [4/11], Loss: nan, Accuracy: 56.97%
Epoch [5/13], Loss: 0.1984, Accuracy: 92.36%
Epoch [5/10], Loss: 0.6145, Accuracy: 68.43%
Epoch [5/8], Loss: 0.1760, Accuracy: 95.93%
Epoch [5/11], Loss: nan, Accuracy: 56.35%
Epoch [6/10], Loss: 0.6099, Accuracy: 69.17%
Epoch [6/13], Loss: 0.1589, Accuracy: 93.59%
Epoch [6/8], Loss: 0.1631, Accuracy: 95.68%
Epoch [6/11], Loss: nan, Accuracy: 61.04%
Epoch [7/10], Loss: 0.6091, Accuracy: 68.80%
Epoch [7/13], Loss: 0.1337, Accuracy: 95.81%
Epoch [7/8], Loss: 0.1400, Accuracy: 97.78%
Epoch [7/11], Loss: nan, Accuracy: 62.15%
Epoch [8/10], Loss: 0.5969, Accuracy: 70.04%
Epoch [8/13], Loss: 0.1200, Accuracy: 95.81%
Epoch [8/8], Loss: 0.1093, Accuracy: 98.52%
Epoch [8/11], Loss: nan, Accuracy: 61.90%
Epoch [9/10], Loss: 0.6065, Accuracy: 69.17%
Epoch [9/13], Loss: 0.1248, Accuracy: 94.94%
🎯 Test Accuracy: 93.39%
[I 2025-03-13 20:20:58,481] Trial 3 finished with value: 93.39080459770115 and param
eters: {'lr': 0.002078800264403543, 'dropout': 0.3082008559678301, 'optimizer': 'Ada
mW', 'num_epochs': 8}. Best is trial 3 with value: 93.39080459770115.
Epoch [9/11], Loss: nan, Accuracy: 64.73%
Epoch [10/13], Loss: 0.1160, Accuracy: 96.42%
Epoch [10/10], Loss: 0.5944, Accuracy: 69.42%
Epoch [1/12], Loss: 0.6273, Accuracy: 64.61%
Epoch [10/11], Loss: nan, Accuracy: 64.24%
Epoch [11/13], Loss: 0.0998, Accuracy: 97.04%
Epoch [2/12], Loss: 0.4380, Accuracy: 84.22%
🎯 Test Accuracy: 78.45%
[I 2025-03-13 20:22:06,941] Trial 1 finished with value: 78.44827586206897 and param
eters: {'lr': 0.002906536218172741, 'dropout': 0.46157561243925405, 'optimizer': 'SG
D', 'num_epochs': 10}. Best is trial 3 with value: 93.39080459770115.
Epoch [11/11], Loss: nan, Accuracy: 62.64%
Epoch [12/13], Loss: 0.1354, Accuracy: 94.70%
Epoch [3/12], Loss: 0.3392, Accuracy: 92.11%
Epoch [1/15], Loss: 0.5089, Accuracy: 73.61%
🎯 Test Accuracy: 75.00%
[I 2025-03-13 20:23:00,191] Trial 2 finished with value: 75.0 and parameters: {'lr':
0.0016296769159054182, 'dropout': 0.454553751910741, 'optimizer': 'SGD', 'num_epoch
```

```
s': 11}. Best is trial 3 with value: 93.39080459770115.
Epoch [13/13], Loss: 0.1355, Accuracy: 95.56%
Epoch [4/12], Loss: 0.2676, Accuracy: 94.08%
Epoch [2/15], Loss: 0.2802, Accuracy: 89.27%
Epoch [1/5], Loss: nan, Accuracy: 57.21%
Epoch [5/12], Loss: 0.2269, Accuracy: 94.33%
🎯 Test Accuracy: 91.67%
[I 2025-03-13 20:23:51,290] Trial 0 finished with value: 91.66666666666666 and param
eters: {'lr': 0.061202104931043756, 'dropout': 0.3178114096276243, 'optimizer': 'SG
D', 'num_epochs': 13}. Best is trial 3 with value: 93.39080459770115.
Epoch [3/15], Loss: 0.1955, Accuracy: 91.37%
Epoch [2/5], Loss: nan, Accuracy: 54.50%
Epoch [6/12], Loss: 0.1954, Accuracy: 95.56%
Epoch [1/13], Loss: 0.7123, Accuracy: 52.90%
Epoch [4/15], Loss: 0.1484, Accuracy: 94.08%
Epoch [3/5], Loss: nan, Accuracy: 61.53%
Epoch [7/12], Loss: 0.1840, Accuracy: 95.68%
Epoch [2/13], Loss: 0.6423, Accuracy: 62.76%
Epoch [5/15], Loss: 0.1525, Accuracy: 94.94%
Epoch [4/5], Loss: nan, Accuracy: 63.87%
Epoch [8/12], Loss: 0.1625, Accuracy: 96.30%
Epoch [6/15], Loss: 0.1178, Accuracy: 95.31%
Epoch [3/13], Loss: 0.6166, Accuracy: 69.30%
Epoch [5/5], Loss: nan, Accuracy: 62.89%
Epoch [9/12], Loss: 0.1642, Accuracy: 97.04%
Epoch [7/15], Loss: 0.0936, Accuracy: 96.92%
Epoch [4/13], Loss: 0.5888, Accuracy: 69.42%
🎯 Test Accuracy: 75.00%
[I 2025-03-13 20:26:37,400] Trial 6 finished with value: 75.0 and parameters: {'lr':
0.0022339987933251227, 'dropout': 0.4637339536492728, 'optimizer': 'SGD', 'num_epoch
s': 5}. Best is trial 3 with value: 93.39080459770115.
Epoch [10/12], Loss: 0.1386, Accuracy: 97.78%
Epoch [8/15], Loss: 0.0698, Accuracy: 97.53%
Epoch [5/13], Loss: 0.5585, Accuracy: 71.52%
Epoch [1/8], Loss: nan, Accuracy: 58.69%
Epoch [11/12], Loss: 0.1621, Accuracy: 96.55%
Epoch [9/15], Loss: 0.0753, Accuracy: 97.16%
Epoch [6/13], Loss: 0.5329, Accuracy: 71.76%
[I 2025-03-13 20:27:24,223] Trial 7 pruned.
Epoch [2/8], Loss: nan, Accuracy: 66.34%
Epoch [12/12], Loss: 0.1509, Accuracy: 97.04%
Epoch [10/15], Loss: 0.0717, Accuracy: 97.16%
Epoch [1/6], Loss: 0.6524, Accuracy: 60.17%
Epoch [3/8], Loss: nan, Accuracy: 71.76%
🎯 Test Accuracy: 93.10%
[I 2025-03-13 20:28:29,938] Trial 4 finished with value: 93.10344827586206 and param
eters: {'lr': 0.001386957211909266, 'dropout': 0.34375037058911473, 'optimizer': 'Ad
amW', 'num_epochs': 12}. Best is trial 3 with value: 93.39080459770115.
Epoch [11/15], Loss: 0.0662, Accuracy: 97.41%
Epoch [2/6], Loss: 0.5140, Accuracy: 76.08%
Epoch [4/8], Loss: nan, Accuracy: 74.11%
[I 2025-03-13 20:29:02,419] Trial 8 pruned.
Epoch [1/7], Loss: 0.5909, Accuracy: 62.76%
Epoch [12/15], Loss: 0.0685, Accuracy: 98.03%
Epoch [3/6], Loss: 0.3718, Accuracy: 82.98%
Epoch [1/5], Loss: nan, Accuracy: 61.90%
```

```
Epoch [2/7], Loss: 0.3215, Accuracy: 87.30%
Epoch [13/15], Loss: 0.0677, Accuracy: 97.16%
Epoch [4/6], Loss: 0.2926, Accuracy: 87.05%
Epoch [2/5], Loss: nan, Accuracy: 77.68%
Epoch [3/7], Loss: 0.2115, Accuracy: 91.37%
Epoch [14/15], Loss: 0.0645, Accuracy: 97.90%
Epoch [5/6], Loss: 0.1906, Accuracy: 92.85%
Epoch [3/5], Loss: nan, Accuracy: 90.14%
Epoch [4/7], Loss: 0.1585, Accuracy: 94.82%
Epoch [15/15], Loss: 0.0843, Accuracy: 97.29%
Epoch [6/6], Loss: 0.1892, Accuracy: 92.85%
[I 2025-03-13 20:30:59,043] Trial 9 pruned.
Epoch [4/5], Loss: nan, Accuracy: 90.14%
Epoch [5/7], Loss: 0.1813, Accuracy: 93.71%
🎯 Test Accuracy: 93.39%
[I 2025-03-13 20:31:30,178] Trial 5 finished with value: 93.39080459770115 and param
eters: {'lr': 0.012751529538156727, 'dropout': 0.40998819876694803, 'optimizer': 'Ad
amW', 'num_epochs': 15}. Best is trial 3 with value: 93.39080459770115.
Epoch [1/9], Loss: 0.6895, Accuracy: 61.16%
[I 2025-03-13 20:31:33,993] Trial 12 pruned.
Epoch [5/5], Loss: nan, Accuracy: 92.23%
[I 2025-03-13 20:32:01,671] Trial 11 pruned.
Epoch [6/7], Loss: 0.1029, Accuracy: 96.30%
Epoch [1/8], Loss: 0.5561, Accuracy: 71.64%
Epoch [1/15], Loss: 0.5237, Accuracy: 73.00%
Epoch [1/15], Loss: nan, Accuracy: 72.01%
Epoch [7/7], Loss: 0.1194, Accuracy: 96.55%
Epoch [2/8], Loss: 0.2846, Accuracy: 90.51%
Epoch [2/15], Loss: 0.2754, Accuracy: 90.63%
🎯 Test Accuracy: 92.82%
[I 2025-03-13 20:33:09,216] Trial 10 finished with value: 92.81609195402298 and para
meters: {'lr': 0.008294039193832786, 'dropout': 0.4871118701458195, 'optimizer': 'Ad
amW', 'num_epochs': 7}. Best is trial 3 with value: 93.39080459770115.
Epoch [2/15], Loss: nan, Accuracy: 89.89%
Epoch [3/8], Loss: 0.1995, Accuracy: 93.34%
Epoch [3/15], Loss: 0.2072, Accuracy: 91.00%
Epoch [1/15], Loss: 0.5942, Accuracy: 66.71%
Epoch [3/15], Loss: nan, Accuracy: 91.86%
Epoch [4/8], Loss: 0.1711, Accuracy: 93.46%
Epoch [4/15], Loss: 0.1903, Accuracy: 93.22%
Epoch [2/15], Loss: 0.3375, Accuracy: 87.42%
Epoch [4/15], Loss: nan, Accuracy: 94.08%
Epoch [5/8], Loss: 0.1584, Accuracy: 95.19%
Epoch [5/15], Loss: 0.1408, Accuracy: 95.68%
Epoch [3/15], Loss: 0.2340, Accuracy: 92.11%
Epoch [5/15], Loss: nan, Accuracy: 93.34%
Epoch [6/8], Loss: 0.1256, Accuracy: 95.68%
Epoch [6/15], Loss: 0.1188, Accuracy: 95.68%
Epoch [4/15], Loss: 0.1790, Accuracy: 93.83%
Epoch [6/15], Loss: nan, Accuracy: 94.08%
[I 2025-03-13 20:35:38,750] Trial 15 pruned.
Epoch [7/8], Loss: 0.0966, Accuracy: 96.92%
Epoch [7/15], Loss: 0.1274, Accuracy: 95.44%
[I 2025-03-13 20:35:52,065] Trial 14 pruned.
Epoch [5/15], Loss: 0.1731, Accuracy: 94.20%
Epoch [1/15], Loss: nan, Accuracy: 72.87%
```

```
Epoch [8/8], Loss: 0.0726, Accuracy: 97.66%
Epoch [1/9], Loss: 0.5699, Accuracy: 68.56%
Epoch [6/15], Loss: 0.1236, Accuracy: 96.67%
Epoch [2/15], Loss: nan, Accuracy: 87.92%
🎯 Test Accuracy: 92.82%
[I 2025-03-13 20:36:56,060] Trial 13 finished with value: 92.81609195402298 and para
meters: {'lr': 0.005167788246562698, 'dropout': 0.36084162703774086, 'optimizer': 'A
damW', 'num_epochs': 8}. Best is trial 3 with value: 93.39080459770115.
Epoch [2/9], Loss: 0.2449, Accuracy: 91.12%
Epoch [7/15], Loss: 0.1224, Accuracy: 96.92%
Epoch [3/15], Loss: nan, Accuracy: 91.37%
Epoch [1/10], Loss: 0.5804, Accuracy: 69.30%
Epoch [3/9], Loss: 0.2109, Accuracy: 91.49%
Epoch [8/15], Loss: 0.0912, Accuracy: 97.41%
Epoch [4/15], Loss: nan, Accuracy: 93.46%
Epoch [2/10], Loss: 0.2860, Accuracy: 88.04%
Epoch [4/9], Loss: 0.1960, Accuracy: 93.09%
[I 2025-03-13 20:38:20,386] Trial 18 pruned.
Epoch [9/15], Loss: 0.0843, Accuracy: 97.90%
Epoch [5/15], Loss: nan, Accuracy: 93.59%
[I 2025-03-13 20:38:51,843] Trial 17 pruned.
Epoch [3/10], Loss: 0.2167, Accuracy: 91.99%
Epoch [1/11], Loss: 0.5470, Accuracy: 70.28%
Epoch [10/15], Loss: 0.0732, Accuracy: 98.03%
Epoch [1/11], Loss: nan, Accuracy: 62.64%
Epoch [4/10], Loss: 0.1598, Accuracy: 93.59%
Epoch [2/11], Loss: 0.3086, Accuracy: 89.15%
Epoch [11/15], Loss: 0.0590, Accuracy: 99.01%
Epoch [2/11], Loss: nan, Accuracy: 72.26%
[I 2025-03-13 20:40:05,316] Trial 21 pruned.
Epoch [5/10], Loss: 0.1677, Accuracy: 93.46%
[I 2025-03-13 20:40:11,902] Trial 19 pruned.
Epoch [3/11], Loss: 0.2429, Accuracy: 92.23%
Epoch [12/15], Loss: 0.0824, Accuracy: 97.66%
Epoch [1/13], Loss: nan, Accuracy: 70.78%
Epoch [1/13], Loss: 0.5620, Accuracy: 65.72%
Epoch [4/11], Loss: 0.1790, Accuracy: 93.83%
Epoch [13/15], Loss: 0.0993, Accuracy: 97.66%
Epoch [2/13], Loss: nan, Accuracy: 88.53%
Epoch [2/13], Loss: 0.3334, Accuracy: 86.31%
Epoch [5/11], Loss: 0.1559, Accuracy: 95.31%
Epoch [14/15], Loss: 0.0703, Accuracy: 98.03%
Epoch [3/13], Loss: nan, Accuracy: 92.48%
Epoch [3/13], Loss: 0.2043, Accuracy: 91.99%
Epoch [6/11], Loss: 0.1216, Accuracy: 96.55%
Epoch [15/15], Loss: 0.0557, Accuracy: 98.64%
Epoch [4/13], Loss: nan, Accuracy: 90.75%
[I 2025-03-13 20:42:28,100] Trial 22 pruned.
Epoch [4/13], Loss: 0.1914, Accuracy: 93.22%
[I 2025-03-13 20:42:34,202] Trial 23 pruned.
Epoch [7/11], Loss: 0.0913, Accuracy: 96.55%
🎯 Test Accuracy: 93.97%
[I 2025-03-13 20:42:57,940] Trial 16 finished with value: 93.96551724137932 and para
meters: {'lr': 0.004268185293191574, 'dropout': 0.37772331610047594, 'optimizer': 'A
damW', 'num_epochs': 15}. Best is trial 16 with value: 93.96551724137932.
Epoch [1/12], Loss: nan, Accuracy: 65.97%
```

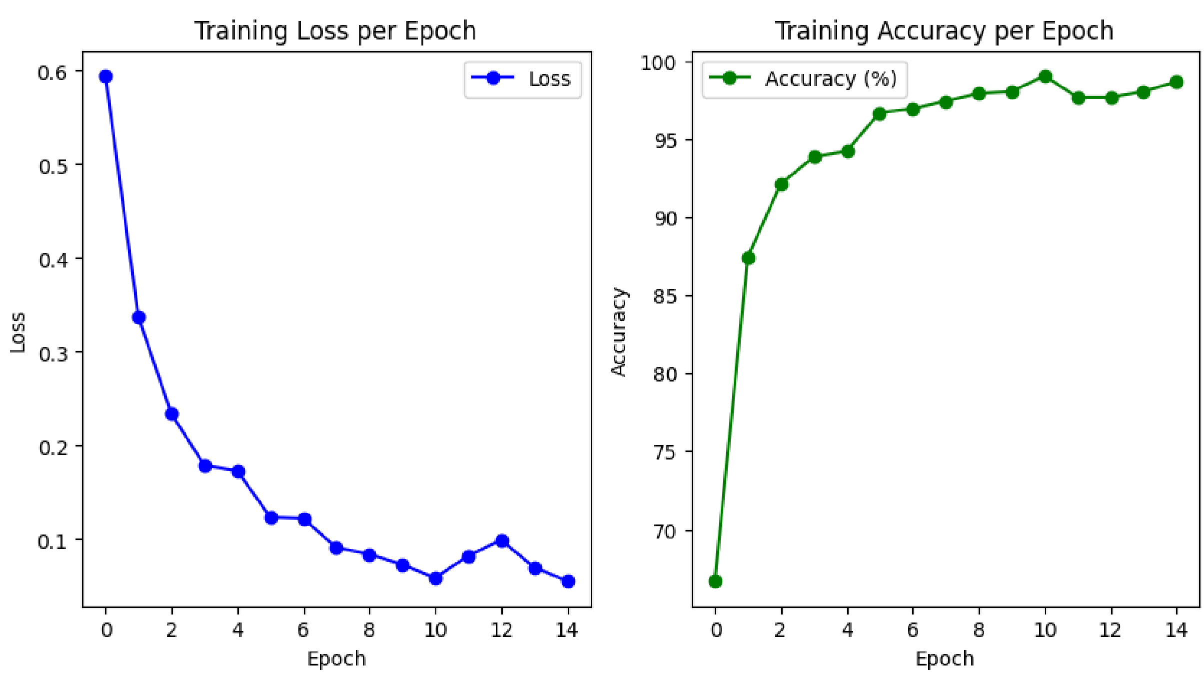
```
Epoch [1/14], Loss: 0.6417, Accuracy: 61.65%
[I 2025-03-13 20:43:09,254] Trial 25 pruned.
Epoch [8/11], Loss: 0.0948, Accuracy: 97.16%
Epoch [1/14], Loss: 0.5463, Accuracy: 72.87%
Epoch [2/12], Loss: nan, Accuracy: 81.75%
[I 2025-03-13 20:43:38,026] Trial 24 pruned.
Epoch [1/14], Loss: 0.6142, Accuracy: 63.26%
Epoch [9/11], Loss: 0.0925, Accuracy: 96.67%
Epoch [2/14], Loss: 0.3222, Accuracy: 90.51%
Epoch [1/14], Loss: nan, Accuracy: 65.10%
Epoch [2/14], Loss: 0.4116, Accuracy: 85.20%
[I 2025-03-13 20:44:19,769] Trial 27 pruned.
Epoch [10/11], Loss: 0.0807, Accuracy: 97.41%
Epoch [3/14], Loss: 0.2245, Accuracy: 93.22%
Epoch [2/14], Loss: nan, Accuracy: 85.45%
[I 2025-03-13 20:44:47,489] Trial 28 pruned.
Epoch [1/14], Loss: 0.6218, Accuracy: 62.39%
Epoch [11/11], Loss: 0.0782, Accuracy: 98.15%
Epoch [4/14], Loss: 0.1784, Accuracy: 94.45%
Epoch [1/7], Loss: nan, Accuracy: 66.95%
Epoch [2/14], Loss: 0.3779, Accuracy: 88.66%
🎯 Test Accuracy: 92.53%
[I 2025-03-13 20:45:34,064] Trial 20 finished with value: 92.52873563218391 and para
meters: {'lr': 0.0035693409643108563, 'dropout': 0.33550898830503706, 'optimizer':
'AdamW', 'num_epochs': 11}. Best is trial 16 with value: 93.96551724137932.
Epoch [5/14], Loss: 0.1439, Accuracy: 95.07%
Epoch [2/7], Loss: nan, Accuracy: 88.66%
Epoch [3/14], Loss: 0.2562, Accuracy: 91.99%
Epoch [1/7], Loss: 0.5219, Accuracy: 74.11%
Epoch [6/14], Loss: 0.1197, Accuracy: 95.68%
Epoch [3/7], Loss: nan, Accuracy: 91.99%
Epoch [4/14], Loss: 0.2019, Accuracy: 93.59%
[I 2025-03-13 20:46:41,852] Trial 29 pruned.
Epoch [2/7], Loss: 0.2312, Accuracy: 90.88%
Epoch [7/14], Loss: 0.0963, Accuracy: 97.04%
Epoch [4/7], Loss: nan, Accuracy: 91.86%
[I 2025-03-13 20:47:06,932] Trial 30 pruned.
Epoch [1/7], Loss: 0.5165, Accuracy: 72.38%
Epoch [3/7], Loss: 0.1824, Accuracy: 91.37%
Epoch [8/14], Loss: 0.0855, Accuracy: 97.53%
Epoch [1/9], Loss: nan, Accuracy: 60.05%
[I 2025-03-13 20:47:45,030] Trial 33 pruned.
Epoch [2/7], Loss: 0.2523, Accuracy: 91.86%
Epoch [4/7], Loss: 0.2164, Accuracy: 92.73%
[I 2025-03-13 20:47:58,335] Trial 31 pruned.
Epoch [9/14], Loss: 0.0765, Accuracy: 97.41%
Epoch [1/12], Loss: nan, Accuracy: 68.19%
Epoch [3/7], Loss: 0.1857, Accuracy: 92.85%
Epoch [1/12], Loss: 0.5856, Accuracy: 70.41%
Epoch [10/14], Loss: 0.0643, Accuracy: 98.52%
Epoch [2/12], Loss: nan, Accuracy: 83.60%
[I 2025-03-13 20:49:01,349] Trial 34 pruned.
Epoch [4/7], Loss: 0.1581, Accuracy: 92.60%
[I 2025-03-13 20:49:11,124] Trial 32 pruned.
Epoch [2/12], Loss: 0.3647, Accuracy: 89.03%
Epoch [11/14], Loss: 0.0764, Accuracy: 98.15%
```

```
Epoch [1/12], Loss: nan, Accuracy: 64.98%
Epoch [1/12], Loss: 0.7157, Accuracy: 49.08%
[I 2025-03-13 20:49:55,120] Trial 37 pruned.
Epoch [3/12], Loss: 0.2639, Accuracy: 93.71%
Epoch [12/14], Loss: 0.0744, Accuracy: 97.90%
Epoch [2/12], Loss: nan, Accuracy: 78.30%
[I 2025-03-13 20:50:23,956] Trial 36 pruned.
Epoch [1/15], Loss: 0.7024, Accuracy: 52.03%
[I 2025-03-13 20:50:35,405] Trial 38 pruned.
Epoch [4/12], Loss: 0.2366, Accuracy: 92.11%
[I 2025-03-13 20:50:39,414] Trial 35 pruned.
Epoch [13/14], Loss: 0.0682, Accuracy: 98.40%
Epoch [1/15], Loss: nan, Accuracy: 50.31%
[I 2025-03-13 20:51:00,579] Trial 39 pruned.
Epoch [1/10], Loss: 0.6817, Accuracy: 57.83%
[I 2025-03-13 20:51:12,135] Trial 40 pruned.
Epoch [1/10], Loss: 0.6917, Accuracy: 53.39%
[I 2025-03-13 20:51:16,000] Trial 41 pruned.
Epoch [14/14], Loss: 0.0767, Accuracy: 97.90%
Epoch [1/10], Loss: nan, Accuracy: 53.27%
[I 2025-03-13 20:51:38,811] Trial 42 pruned.
Epoch [1/13], Loss: 0.5765, Accuracy: 68.80%
Epoch [1/6], Loss: 0.5401, Accuracy: 71.52%
🎯 Test Accuracy: 90.80%
[I 2025-03-13 20:52:15,089] Trial 26 finished with value: 90.80459770114942 and para
meters: {'lr': 0.0031561675802634355, 'dropout': 0.34646203283673693, 'optimizer':
'AdamW', 'num_epochs': 14}. Best is trial 16 with value: 93.96551724137932.
Epoch [1/6], Loss: nan, Accuracy: 68.19%
Epoch [2/13], Loss: 0.3581, Accuracy: 88.78%
Epoch [2/6], Loss: 0.2682, Accuracy: 91.25%
Epoch [1/6], Loss: 0.5683, Accuracy: 70.65%
Epoch [2/6], Loss: nan, Accuracy: 89.40%
Epoch [3/13], Loss: 0.2441, Accuracy: 93.59%
Epoch [3/6], Loss: 0.2128, Accuracy: 92.36%
Epoch [2/6], Loss: 0.2781, Accuracy: 91.25%
Epoch [3/6], Loss: nan, Accuracy: 91.25%
[I 2025-03-13 20:53:46,785] Trial 45 pruned.
Epoch [4/13], Loss: 0.2140, Accuracy: 93.59%
[I 2025-03-13 20:54:02,750] Trial 43 pruned.
Epoch [4/6], Loss: 0.2000, Accuracy: 92.60%
[I 2025-03-13 20:54:08,850] Trial 44 pruned.
Epoch [3/6], Loss: 0.1858, Accuracy: 93.71%
Epoch [1/8], Loss: nan, Accuracy: 69.42%
Epoch [1/8], Loss: 0.6054, Accuracy: 64.49%
Epoch [1/8], Loss: 0.6463, Accuracy: 62.39%
[I 2025-03-13 20:54:47,504] Trial 49 pruned.
Epoch [4/6], Loss: 0.1475, Accuracy: 94.20%
Epoch [2/8], Loss: nan, Accuracy: 87.55%
Epoch [2/8], Loss: 0.3371, Accuracy: 87.67%
Epoch [5/6], Loss: 0.1374, Accuracy: 95.31%
Epoch [3/8], Loss: nan, Accuracy: 90.63%
[I 2025-03-13 20:55:43,164] Trial 47 pruned.
Epoch [3/8], Loss: 0.2571, Accuracy: 92.23%
Epoch [6/6], Loss: 0.1626, Accuracy: 94.94%
[I 2025-03-13 20:56:11,607] Trial 46 pruned.
Epoch [4/8], Loss: 0.1731, Accuracy: 93.46%
```

```
[I 2025-03-13 20:56:27,584] Trial 48 pruned.

Best Hyperparameters: {'lr': 0.004268185293191574, 'dropout': 0.37772331610047594,
'optimizer': 'AdamW', 'num_epochs': 15}

Accuracy: 93.96551724137932
```

In [7]:

```python
best_trial = study.best_trial
best_train_losses = best_trial.user_attrs.get("train_losses", [])
best_train_accuracies = best_trial.user_attrs.get("train_accuracies", [])

plt.figure(figsize=(10, 5))

# Loss Graph
plt.subplot(1, 2, 1)
plt.plot(best_train_losses, label="Loss", marker="o", linestyle="-", color="b")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training Loss per Epoch")
plt.legend()

# Accuracy Graph
plt.subplot(1, 2, 2)
plt.plot(best_train_accuracies, label="Accuracy (%)", marker="o", linestyle="-", co
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Training Accuracy per Epoch")
plt.legend()

plt.show()
```

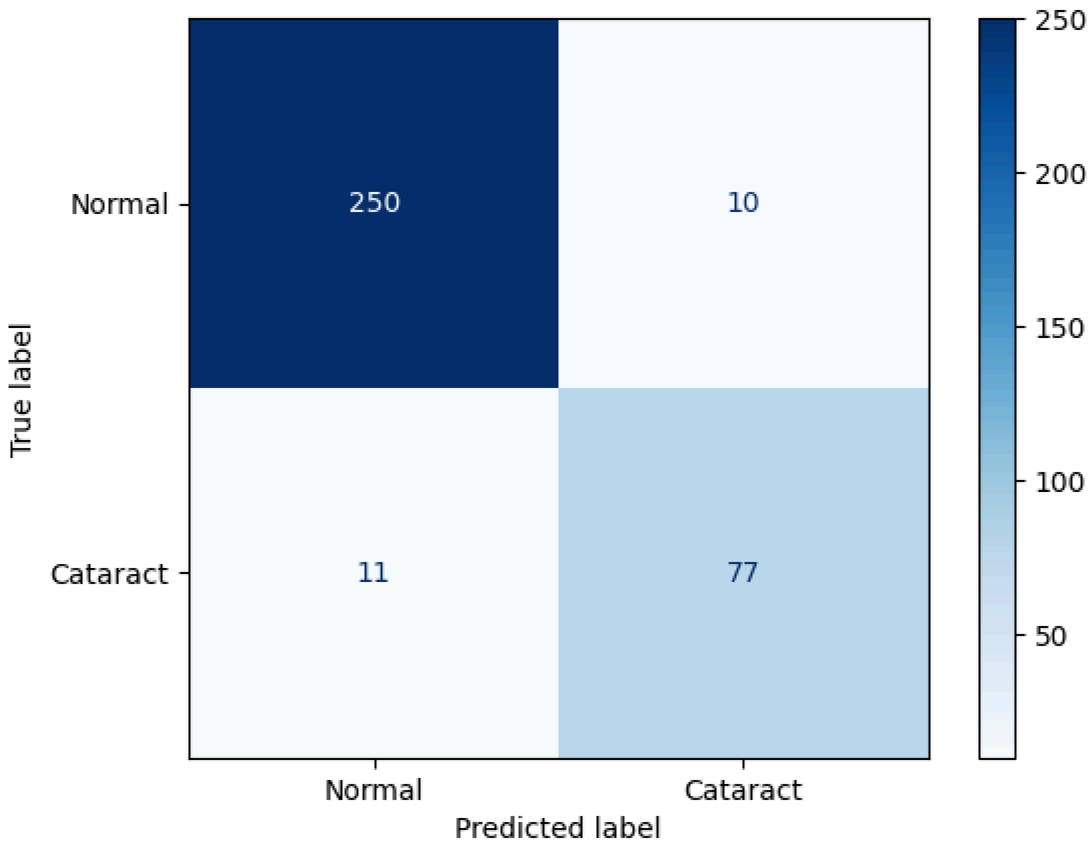


In [8]:

```python
best_preds = np.array(study.user_attrs.get("best_preds", []))
best_labels = np.array(study.user_attrs.get("best_labels", []))

conf_matrix = confusion_matrix(best_preds, best_labels)
```

```
ConfusionMatrixDisplay(conf_matrix, display_labels=class_labels).plot(cmap=plt.cm.B
cr = classification_report(best_preds, best_labels)

print(f"Accuracy: {acc:.2f}%")
print(cr)
```

```
Accuracy: 93.97%
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96       260
         1.0       0.89      0.88      0.88        88

    accuracy                           0.94       348
   macro avg       0.92      0.92      0.92       348
weighted avg       0.94      0.94      0.94       348
```



In [9]:
```
# # save
output_model_path = f"output/checkpoint-{acc}-hyperparam.pth"

torch.save({
    "model_state_dict": study.user_attrs.get("best_model_state"),  # Best model wei
    "optimizer_state_dict": study.user_attrs.get("best_optimizer_state"),  # Best o
    "best_hyperparameters": study.best_params,
    "best_accuracy": study.best_value,
    "best_train_losses": study.user_attrs.get("best_train_losses", []),  # Best tra
    "best_train_accuracies": study.user_attrs.get("best_train_accuracies", []),  #
}, output_model_path)
```

In [10]:

```
# checkpoint = torch.load("checkpoint.pth")
# model.load_state_dict(checkpoint['model_state_dict'])
# optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
# model.eval()
```