# Explanation of Testing Phase with User Data (test_system)

This section explains the function used to test the system's models on both predefined and user-provided data.

```
def test_system(tree, user_image_folder=None, user_text_csv=None):
    print("
=== Testing phase ===")
    results = {}

    # Test on predefined datasets
    for node_id in tree.models:
        dataset_name = tree.graph.nodes[node_id]["dataset"]
        data_loader = tree.data_distribution[node_id]["loader"]
        data_type = tree.data_distribution[node_id]["type"]
        accuracy = evaluate_model(tree.models[node_id], data_loader, data_type)
        results[node_id] = {"dataset": dataset_name, "accuracy": accuracy}
        print(f"{node_id} ({dataset_name}) Accuracy: {accuracy:.2f}%")

    # Test on User provided data
    user_loaders = []
    if user_image_folder:
        user_loader, sample, data_type, dataset_name =
load_user_data(image_folder=user_image_folder)
        user_loaders.append((user_loader, user_sample, data_type, dataset_name, "user_image"))
    if user_text_csv:
        user_loader, sample, data_type, dataset_name = load_user_data(text_csv=user_text_csv)
        user_loaders.append((user_loader, user_sample, data_type, dataset_name, "user_text"))
    for loader, sample, data_type, dataset_name, node_id in user_loaders:
        parent_accuracy = evaluate_model(tree.models["parent"], loader, data_type)
        print(f"Parent Accuracy on {dataset_name}: {parent_accuracy:.2f}%")
        if spawn_child_models(tree.models["parent"], loader, sample, dataset_name, data_type,
node_id, tree):
            accuracy = evaluate_model(tree.models[node_id], loader, data_type)
            results[node_id] = {"dataset": dataset_name, "accuracy": accuracy}
            print(f"{node_id} ({dataset_name}) Accuracy: {accuracy:.2f}%")
        else:
            results[node_id] = {"dataset": dataset_name, "accuracy": parent_accuracy}

    # Collective Performance
    avg_accuracy = np.mean([r["accuracy"] for r in results.values() if r["accuracy"] > 0])
    print(f"System Average Accuracy: {avg_accuracy:.2f}%")

    # Test Knowledge Transfer for user text data (if applicable)
    if user_text_csv and "user_text" in tree.models:
        print("
Testing User Text without Knowledge Transfer...")
        no_transfer_model = ChildRNN(input_size=len(vocab), hidden_size=128,
output_size=2).to(device)
        optimizer = optim.Adam(no_transfer_model.parameters(), lr=0.001)
        for epoch in range(2):
            no_transfer_model.train()
            for data, target in loader:
                data, target = data.to(device), target.to(device)
                data = data.long()
                optimizer.zero_grad()
                output = no_transfer_model(data)
                loss = criterion(output, target)
                loss.backward()
                optimizer.step()
            no_transfer_accuracy = evaluate_model(no_transfer_model, loader, "sequence")
            print(f"User Text No-Transfer Model Accuracy: {no_transfer_accuracy:.2f}%")
            print(f"Knowledge Transfer Benefit: {results['user_text']['accuracy'] - no_transfer_accuracy:.2f}%")

    # Prune Underperforming nodes
    for node_id in list(tree.models.keys()):
        if node_id != "parent" and tree.prune_node(node_id, min_accuracy=50.0):
            print(f"Pruned {node_id} due to low performance.")
    return results
```

Line-by-line explanation:

- Evaluates all models in the tree on their respective datasets and prints accuracy.
- Loads and tests user-provided image and text data, spawns child models if needed.
- Calculates and prints system average accuracy.
- Optionally tests knowledge transfer for user text data by training a model without transfer and comparing accuracy.
- Prunes underperforming child models from the tree.
- Returns a dictionary of results.

Purpose:

- This function provides a comprehensive evaluation of the system's performance on both standard and user data,
  supports knowledge transfer analysis, and manages model pruning.