Low Level Design Document

This Low Level Design (LLD) document outlines the core implementation details for the **TitanicTree - Survival Prediction Web App**. The application predicts Titanic passenger survival using a Decision Tree model, provides data upload, model visualization, and evaluation metrics, and is deployable via Docker.

---

1. System Components

Component	Description	Key Responsibilities
Web UI	Flask-based frontend	Data upload, results display, user input
API Layer	Flask routes/controllers	Handle requests, trigger ML logic
ML Service	Decision Tree logic (scikit-learn)	Train, predict, evaluate, visualize model
Data Handler	Data preprocessing (pandas, numpy)	Clean, validate, transform input data
Docker Container	Deployment environment	Encapsulate app and dependencies

---

2. Class/Interface Overview

Class/Module	Description	Key Methods/Attributes
`app.py`	Flask app entrypoint	`@app.route` handlers
`MLService`	ML logic encapsulation	`train()`, `predict()`, `evaluate()`, `visualize_tree()`
`DataHandler`	Data loading & preprocessing	`load_data()`, `clean_data()`, `transform()`
`templates/`	HTML templates (Jinja2)	N/A
`static/`	Static files (CSS, images)	N/A

Key Method Example:

```
class MLService:
    def train(self, data): ...
    def predict(self, input): ...
    def evaluate(self, test_data): ...
    def visualize_tree(self): ...
```

---

3. Data Structure Overview

Model/Schema	Fields/Attributes

Passenger	`Pclass`, `Sex`, `Age`, `SibSp`, `Parch`, `Fare`, `Embarked`, `Survived`
PredictionInput	Subset of Passenger fields (excluding `Survived`)
EvaluationMetrics	`accuracy`, `precision`, `recall`, `f1_score`

---

4. Algorithms/Logic

Prediction Flow (Pseudocode):

```
# On data upload:
data = DataHandler.load_data(file)
cleaned = DataHandler.clean_data(data)
X_train, X_test, y_train, y_test = split(cleaned)
MLService.train(X_train, y_train)
metrics = MLService.evaluate(X_test, y_test)
tree_img = MLService.visualize_tree()

# On prediction:
input = get_user_input()
result = MLService.predict(input)
```

---

5. Error Handling

Scenario	Handling Approach
Invalid file upload	Return error message to user
Data format/schema mismatch	Validate & show descriptive error
Model training failure	Log error, notify user
Prediction on incomplete input	Prompt user to complete required fields
Internal server error	Return generic error, log details

End of Document