Introduction

- **Purpose:** Deploy a prediction model for identifying late shipments.
- **Goals:** Provide accurate predictions to improve logistics planning.
- **Scope:** Deployment on Google Cloud Platform (GCP) using managed services.

Architecture

Components:

- Trained prediction model serialized using joblib or pickle.
- Google Cloud Storage (GCS) for storing serialized model file.
- Google Cloud Functions or AI Platform Prediction for serving prediction API.
- Google Cloud Endpoints for API gateway and security.

Interactions:

- Clients send shipment data to prediction API endpoint.
- Model performs inference and returns prediction results.

Design Decisions

- **Serialization:** Choose joblib or pickle for model serialization based on compatibility with GCP services.
- **Deployment Method:** Selected Google Cloud AI Platform for managed deployment and scalability.

Deployment

- **Infrastructure:** Use GCP resources such as Cloud Storage, Cloud Functions, and AI Platform Prediction.
- **Deployment Process:** Upload serialized model to GCS, deploy Cloud Function or AI Platform model version.
- **Scalability:** Utilize managed services for auto-scaling based on demand.

Security

- **Authentication:** Configure API gateway with authentication using API keys or OAuth 2.0.
- **Authorization:** Define access control policies to restrict API access based on user roles.
- **Data Protection:** Encrypt sensitive data in transit and at rest using GCP encryption services.

Performance

- **Requirements:** Ensure low-latency responses for prediction API.
- **Optimizations:** Optimize model inference speed, use caching for frequently accessed data.
- **Scalability Strategies:** Utilize auto-scaling features of GCP services to handle varying loads.

Monitoring and Logging

- **Monitoring:** Set up Google Cloud Monitoring for tracking API metrics (latency, error rates).
- **Logging:** Use Google Cloud Logging to collect logs for monitoring system behaviour and diagnosing issues.

Testing

- **Unit Tests:** Develop unit tests to validate individual components (serialization, API endpoints).
- **Integration Tests:** Test end-to-end functionality of prediction API with sample data.
- **Performance Testing:** Evaluate API performance under different load conditions to ensure scalability.

Maintenance and Support

- **Updates:** Plan for regular updates to the prediction model and API endpoints.
- **Maintenance:** Monitor system health and apply patches as needed to ensure reliability.
- **Support:** Provide documentation and support channels for users integrating with the API.