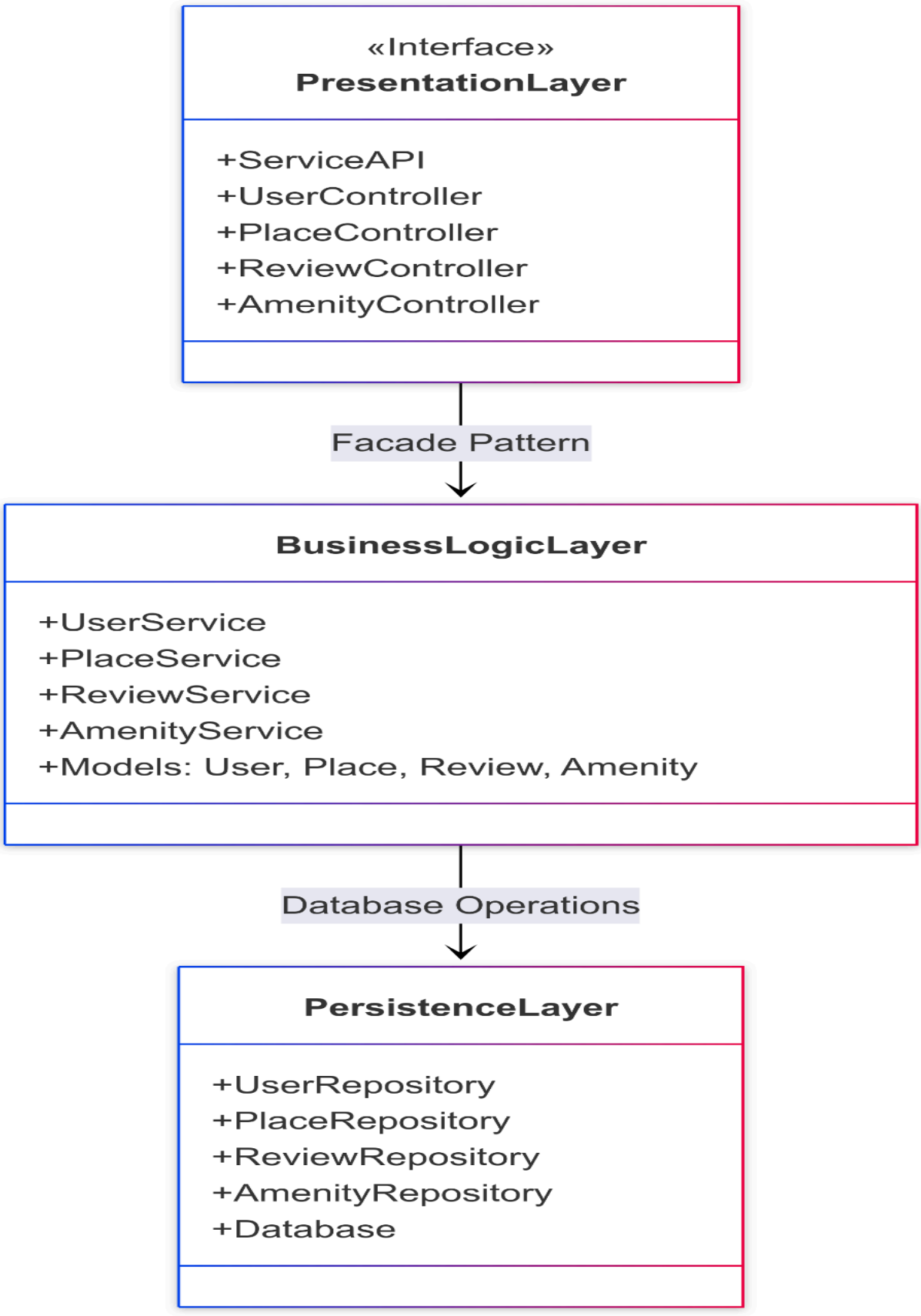
Technical Documentation for HBnB Evolution

Introduction

The **HBnB Evolution** application is a simplified, Airbnb-like platform designed to manage users, property listings (places), reviews, and amenities. This document compiles the foundational technical architecture and design for Part 1 of the project, providing diagrams and explanatory notes that will guide the implementation phase.

High-Level Architecture

Shows the three-layered structure of the application and how they interact via the Facade Pattern.

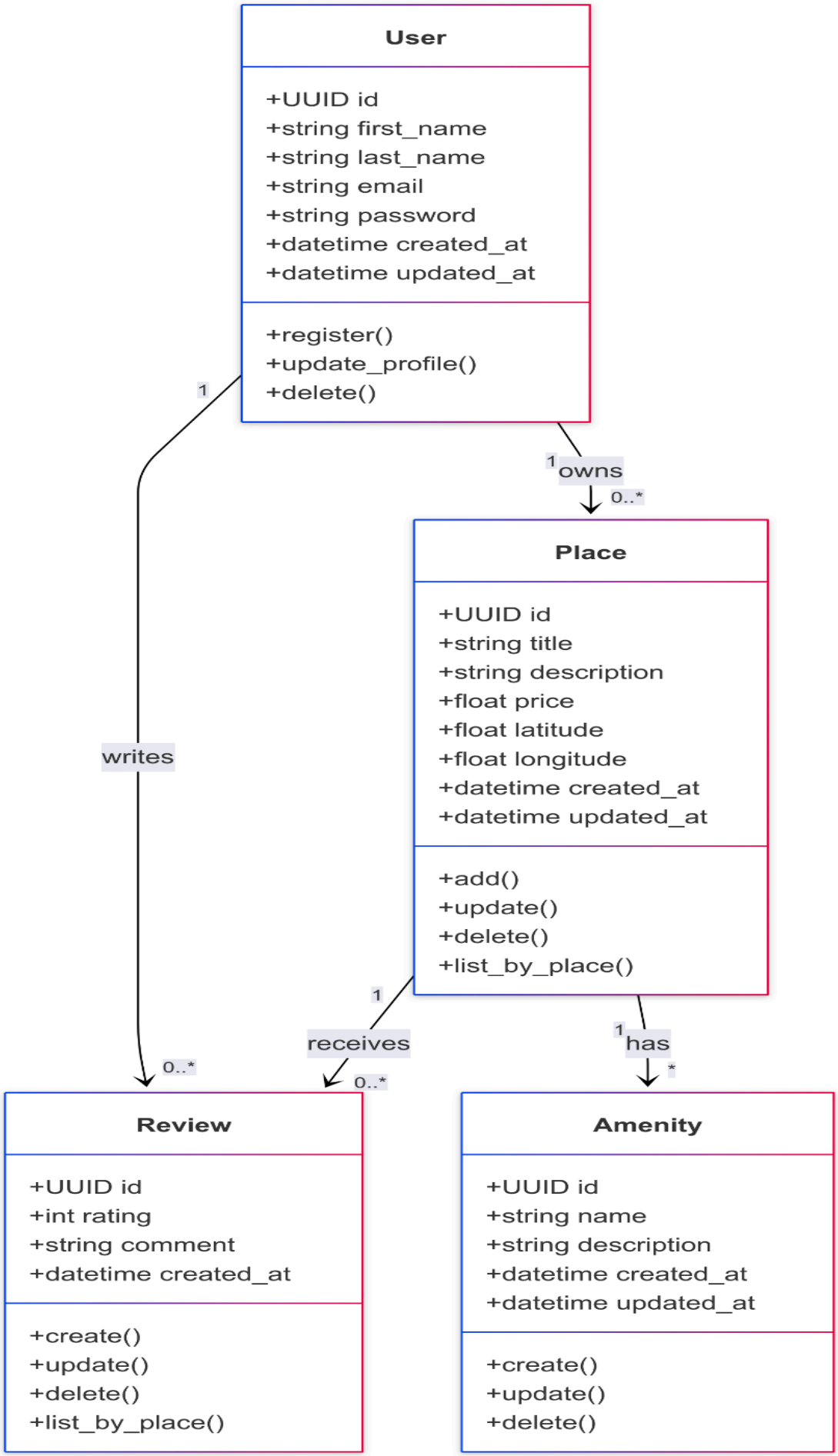

«Interface»
PresentationLayer
+ServiceAPI
+UserController
+PlaceController
+ReviewController
+AmenityController
Facade Pattern
BusinessLogicLayer
+UserService
+PlaceService
+ReviewService
+AmenityService
+Models: User, Place, Review, Amenity
Database Operations
PersistenceLayer
+UserRepository
+PlaceRepository
+ReviewRepository
+AmenityRepository
+Database

Explanatory Notes:

- Presentation Layer: Entry point for client interactions; exposes RESTful endpoints via controllers that delegate to services.

- Business Logic Layer: Implements domain rules in service classes and data models; acts as the facade between controllers and repositories.

- Persistence Layer: Handles CRUD operations against the database through repository classes.


User
+UUID id
+string first_name
+string last_name
+string email
+string password
+datetime created_at
+datetime updated_at
+register()
+update_profile()
+delete()
1
writes
0..*
1
owns
0..*
Place
+UUID id
+string title
+string description
+float price
+float latitude
+float longitude
+datetime created_at
+datetime updated_at
+add()
+update()
+delete()
+list_by_place()
1
receives
0..*
1
has
*
Review
+UUID id
+int rating
+string comment
+datetime created_at
+create()
+update()
+delete()
+list_by_place()
Amenity
+UUID id
+string name
+string description
+datetime created_at
+datetime updated_at
+create()
+update()
+delete()

Business Logic Layer Design

Detailed class diagram depicting the core entities, their attributes, methods, and relationships.

Explanatory Notes:

- **User**: Represents application users; includes registration, profile update, and deletion methods.

- **Place**: Property listings owned by users; supports CRUD operations and amenity listing.

- **Review**: Feedback left by users for places; includes rating and comment management.

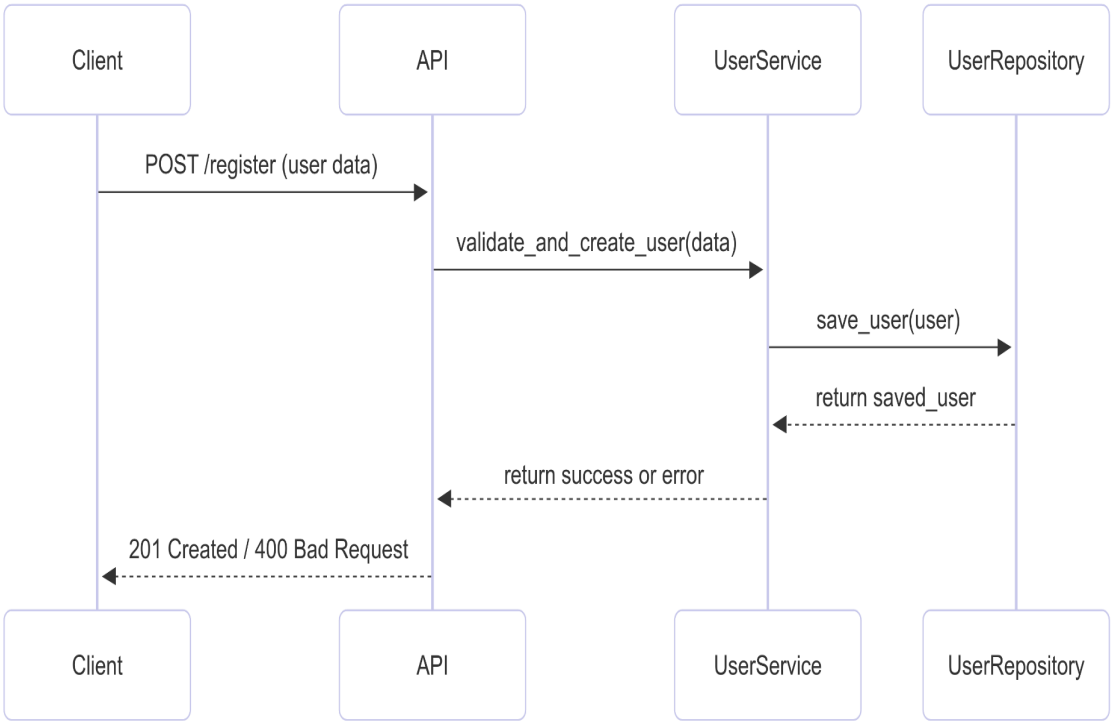
- **Amenity**: Features associated with places; managed independently but linked to places.

- **Relationships**:
    - One-to-many: User→Place, User→Review, Place→Review.
    - Many-to-many (modeled as one-to-many in code): Place→Amenity.

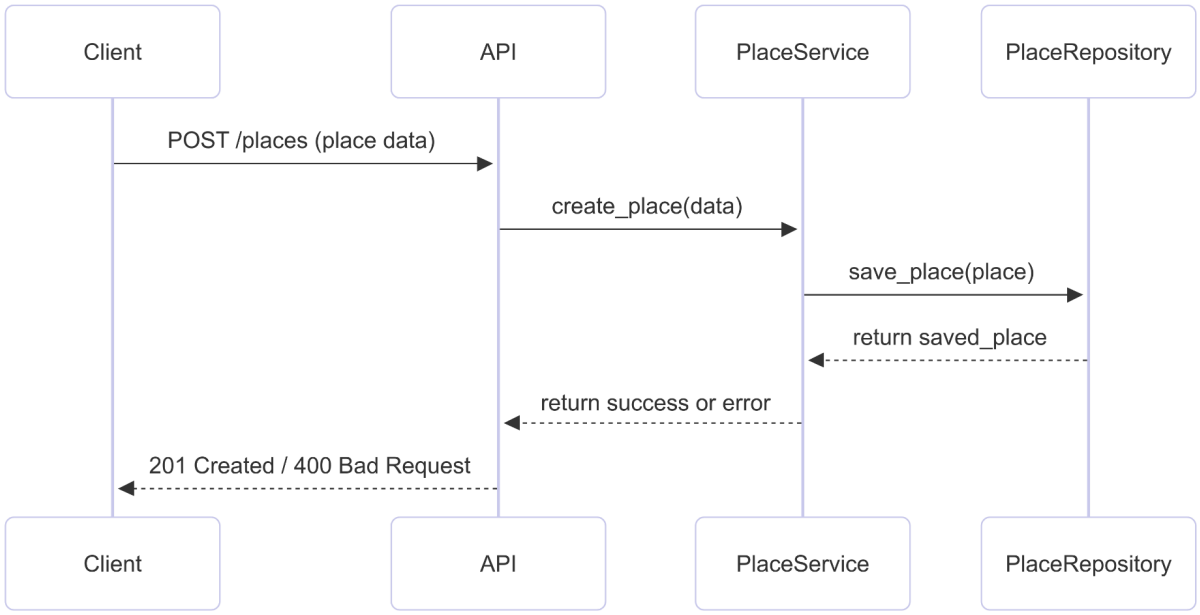
API Interaction Flow

Sequence diagrams for key API calls, illustrating communication across layers.

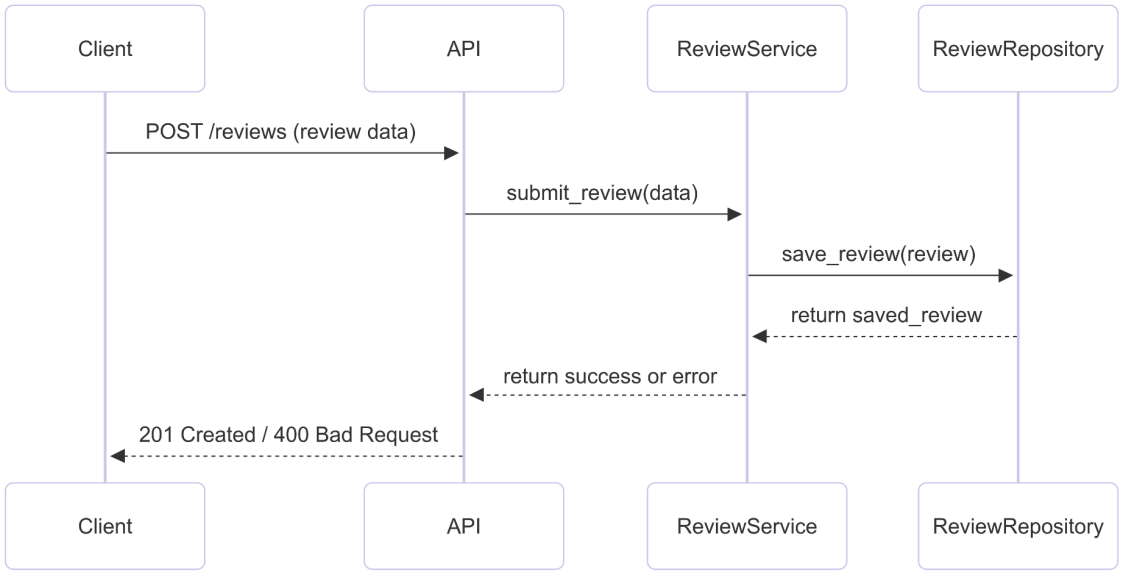
1. User Registration



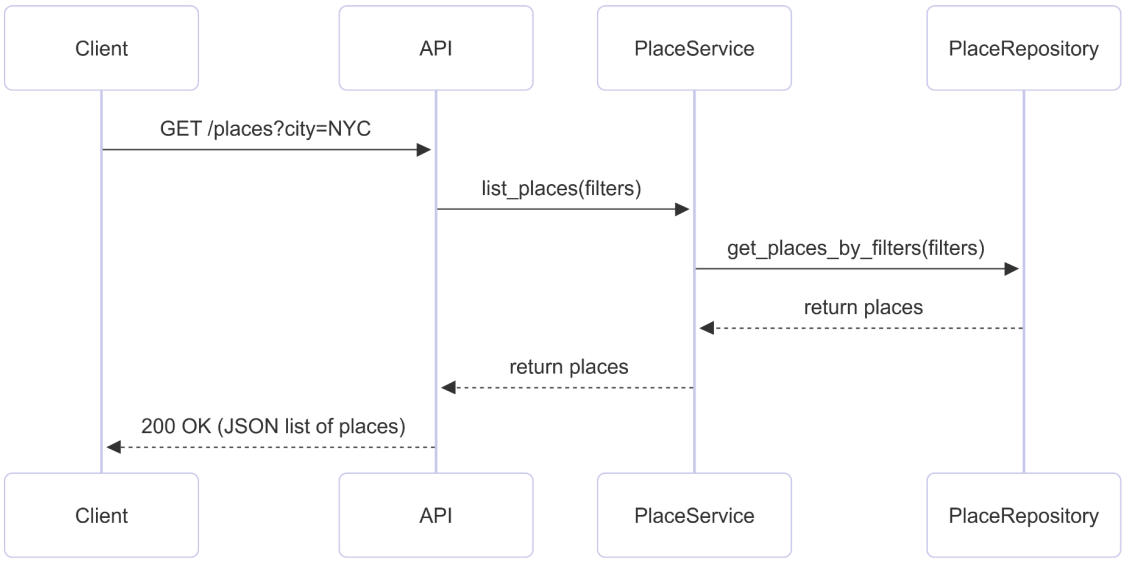
2. Place Creation



3. Review Submission



4. Fetching a List of Places



Explanatory Notes:

- **User Registration**: Validates input, persists user, and returns appropriate HTTP status.
- **Place Creation**: Handles listing details through service and repository layers.
- **Review Submission**: Persists user feedback and returns confirmation.
- **Fetching Places**: Applies filters at service layer and retrieves data from the repository.

Conclusion

This document provides a clear blueprint for HBnB Evolution's architecture and design. The diagrams and explanations serve as a guiding reference for the implementation phases, ensuring consistency with the defined business rules and technical requirements.