Traveller Feedback System

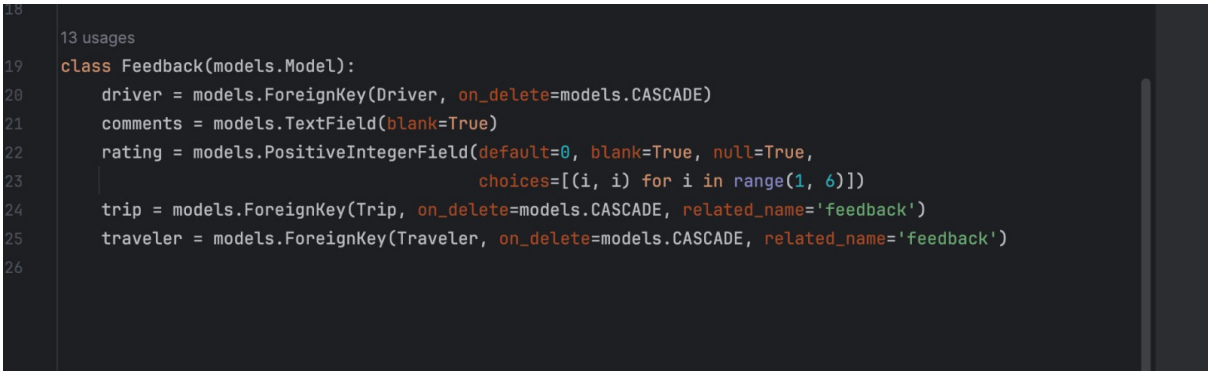
-Implemented in two languages independently (C++ and Python)

Django implementation –

Models

Created four models
1-Driver
2-Traveller
3-Feedback
4-Trip

```python
from django.db import models


9 usages
class Driver(models.Model):
    name = models.CharField(max_length=255)
    age = models.IntegerField


10 usages
class Traveler(models.Model):
    name = models.CharField(max_length=255)
    trips = models.ManyToManyField( to: 'Trip', related_name='travelers')


8 usages
class Trip(models.Model):
    driver = models.ForeignKey(Driver, on_delete=models.CASCADE, related_name='trips')
```



```python
13 usages
class Feedback(models.Model):
    driver = models.ForeignKey(Driver, on_delete=models.CASCADE)
    comments = models.TextField(blank=True)
    rating = models.PositiveIntegerField(default=0, blank=True, null=True,
                                         choices=[(i, i) for i in range(1, 6)])
    trip = models.ForeignKey(Trip, on_delete=models.CASCADE, related_name='feedback')
    traveler = models.ForeignKey(Traveler, on_delete=models.CASCADE, related_name='feedback')
```

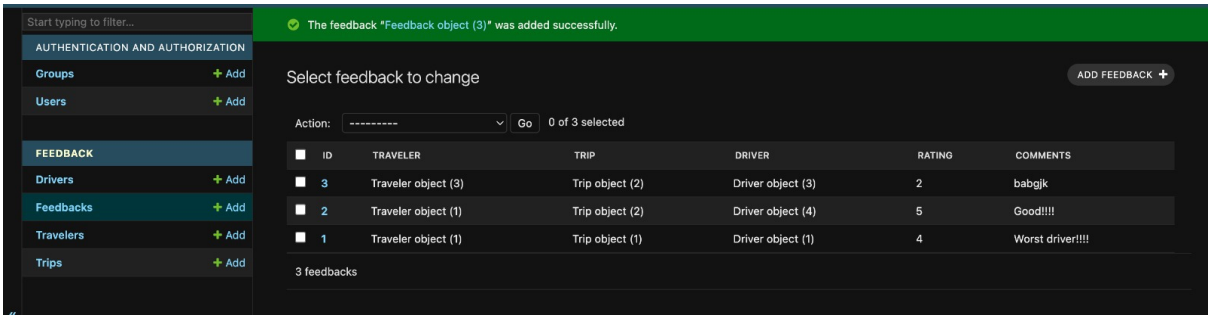
Views -

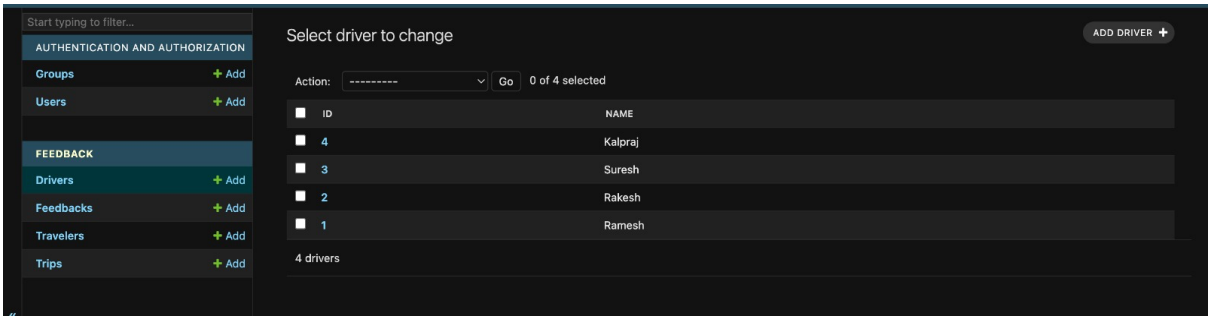
```
2 usages
class FeedbackSubmissionView(generics.CreateAPIView):
    serializer_class = FeedbackSerializer

    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)

        traveler_id = serializer.validated_data['traveler']
        trip_id = serializer.validated_data['trip']
        driver_id = serializer.validated_data['driver']

        try:
            traveler = Traveler.objects.get(id=traveler_id)
            trip = Trip.objects.get(id=trip_id)
            driver = Driver.objects.get(id=driver_id)
        except (Traveler.DoesNotExist, Trip.DoesNotExist, Driver.DoesNotExist):
            return Response( data: {'detail': 'Invalid traveler, trip, or driver ID.'}, status=status.HTTP_400_BAD_REQUEST)


        feedback_data = {
            'traveler': traveler.id,
            'trip': trip.id,
            'driver': driver.id,
            'rating': serializer.validated_data.get('rating', 0),
            'comments': serializer.validated_data.get('comments', ''),
        }

        feedback_serializer = FeedbackSerializer(data=feedback_data)
        feedback_serializer.is_valid(raise_exception=True)
        feedback_serializer.save()

        return Response( data: {'detail': 'Feedback submitted successfully.'}, status=status.HTTP_201_CREATED)
```

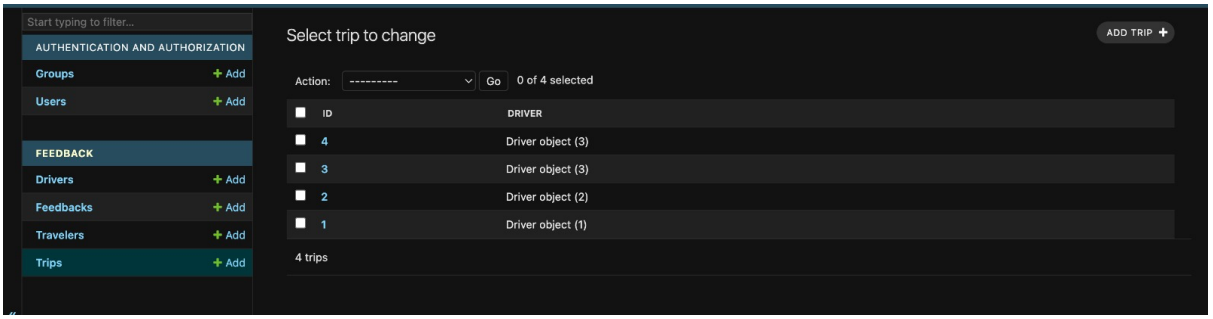
```python
class FeedbackListAllView(generics.ListAPIView):
    queryset = Feedback.objects.all()
    serializer_class = FeedbackSerializer


2 usages
class FeedbackListByTravelerView(generics.ListAPIView):
    serializer_class = FeedbackSerializer

    def get_queryset(self):
        traveler_id = self.kwargs['traveler_id']
        return Feedback.objects.filter(traveler__id=traveler_id)


2 usages
class FeedbackListByDriverView(generics.ListAPIView):
    serializer_class = FeedbackSerializer

    def get_queryset(self):
        driver_id = self.kwargs['driver_id']
        return Feedback.objects.filter(driver__id=driver_id)


2 usages
class FeedbackListByTripView(generics.ListAPIView):
    serializer_class = FeedbackSerializer

    def get_queryset(self):
        trip_id = self.kwargs['trip_id']
        return Feedback.objects.filter(trip__id=trip_id)
```

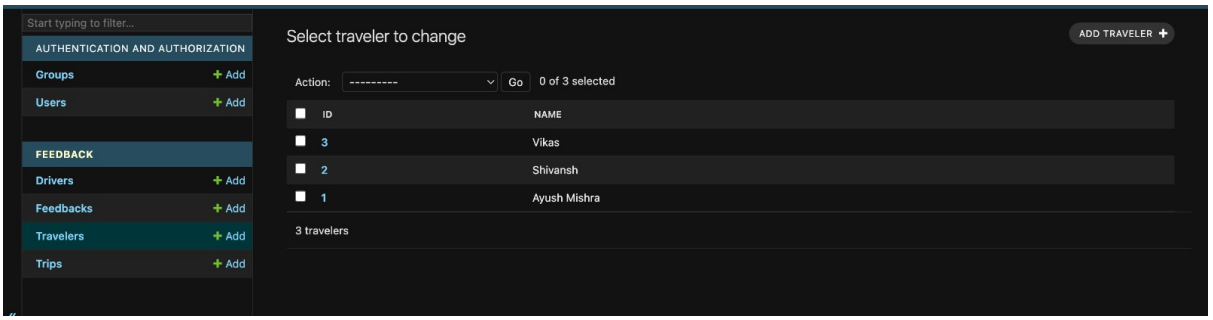
```python
2 usages
class TravelerFeedbackView(generics.ListAPIView):
    serializer_class = FeedbackSerializer

    def get_queryset(self):
        traveler_id = self.kwargs['traveler_id']  # Assuming you have 'traveler_id' in your URL
        return Feedback.objects.filter(traveler__id=traveler_id)


2 usages
class TripOverallFeedbackView(generics.RetrieveAPIView):
    serializer_class = FeedbackSerializer

    def get_object(self):
        trip_id = self.kwargs['trip_id']  # Assuming you have 'trip_id' in your URL
        return Feedback.objects.filter(trip__id=trip_id).aggregate(Avg('rating'))


2 usages
class AggregatedFeedbackView(generics.ListAPIView):
    serializer_class = FeedbackSerializer

    def get_queryset(self):
        return Feedback.objects.values('trip').annotate(average_rating=Avg('rating'))


2 usages
class DriverOverallFeedbackView(generics.RetrieveAPIView):
    serializer_class = FeedbackSerializer

    def get_object(self):
        driver_id = self.kwargs['driver_id']  # Assuming you have 'driver_id' in your URL
        return Feedback.objects.filter(driver__id=driver_id).aggregate(Avg('rating'))
```

Django Admin -

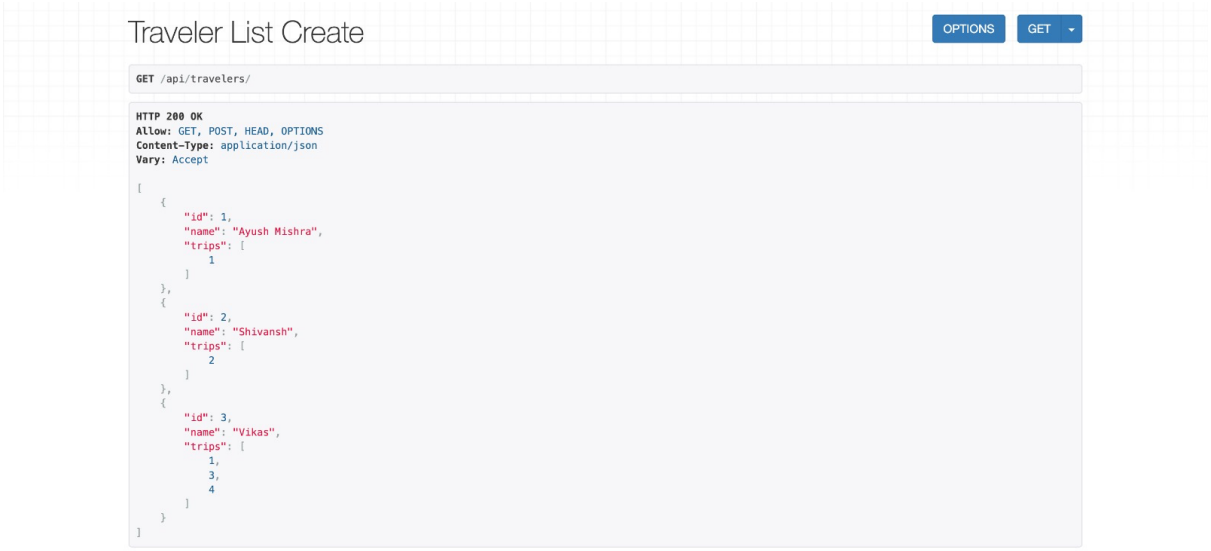


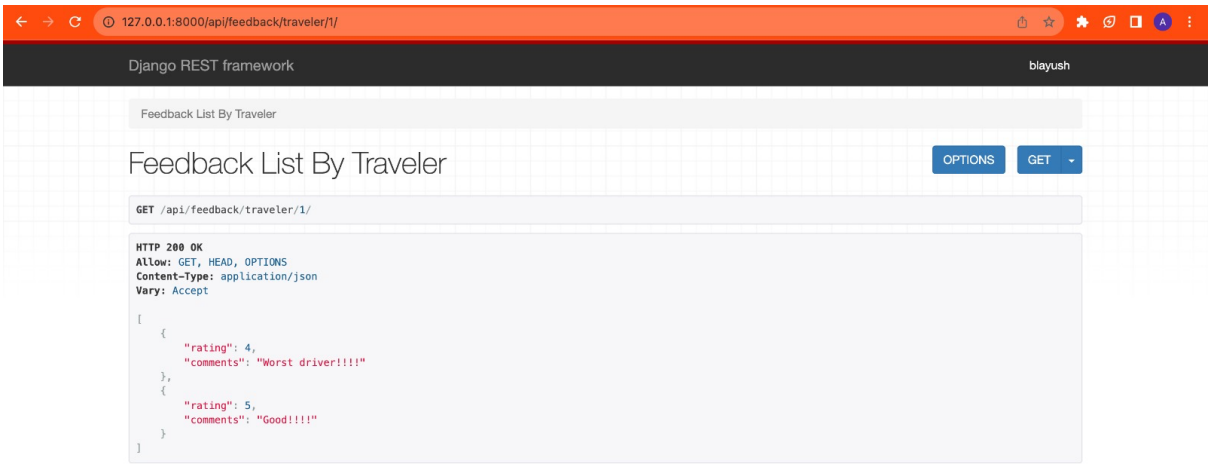


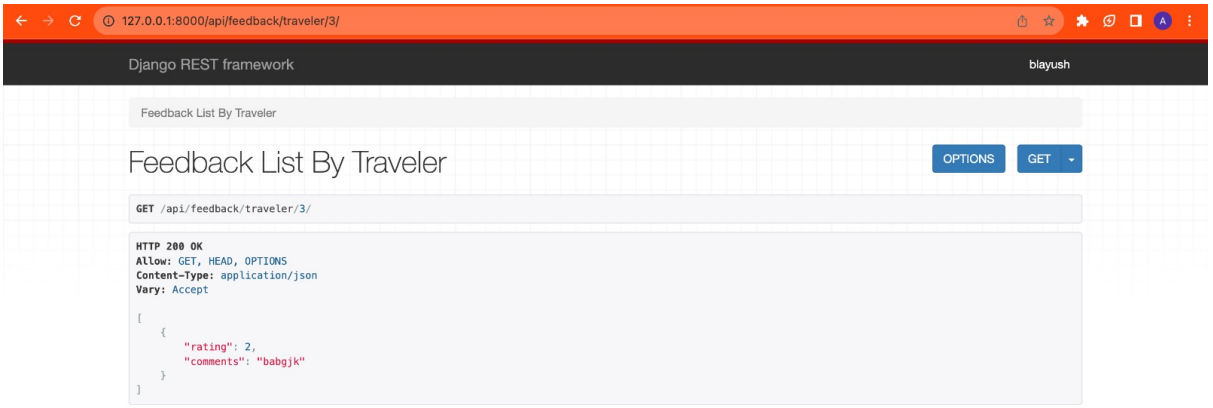


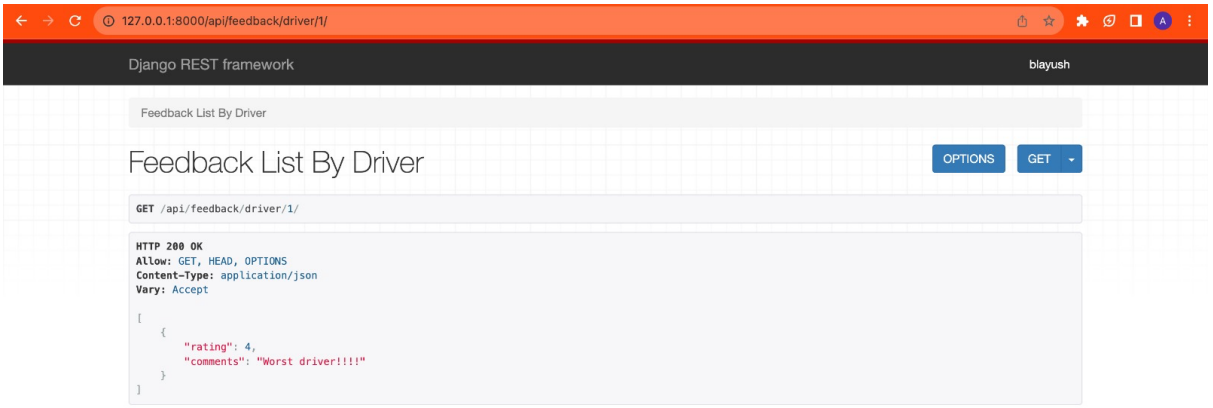


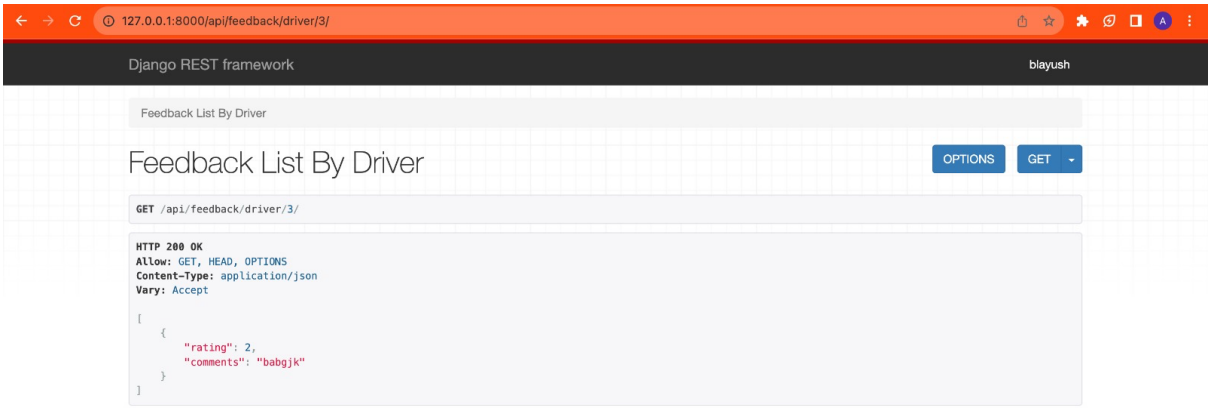
Endpoints -

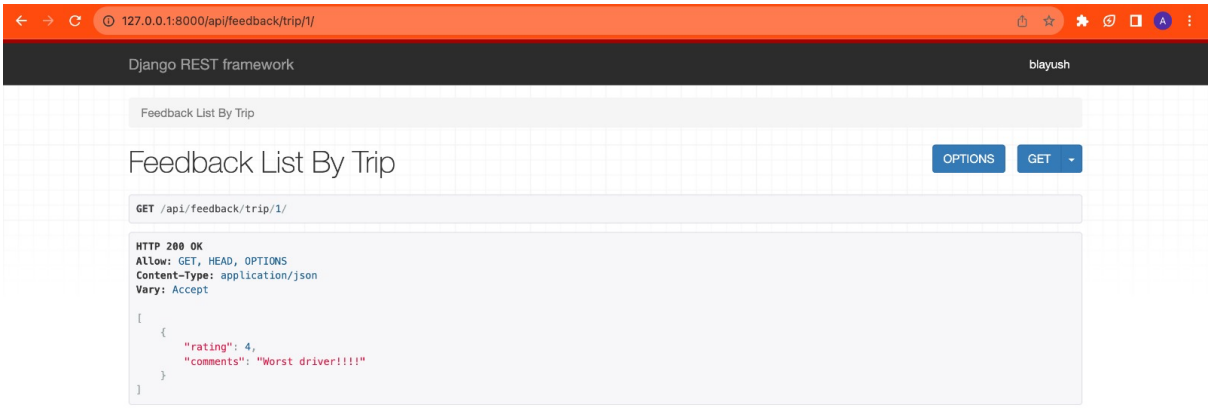

Traveler List Create
OPTIONS
GET
GET /api/travelers/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"name": "Ayush Mishra",
"trips": [
1
]
},
{
"id": 2,
"name": "Shivansh",
"trips": [
2
]
},
{
"id": 3,
"name": "Vikas",
"trips": [
1,
3,
4
]
}
]


127.0.0.1:8000/api/feedback/traveler/1/
Django REST framework
blayush
Feedback List By Traveler
Feedback List By Traveler
OPTIONS
GET
GET /api/feedback/traveler/1/
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"rating": 4,
"comments": "Worst driver!!!!"
},
{
"rating": 5,
"comments": "Good!!!!"
}
]

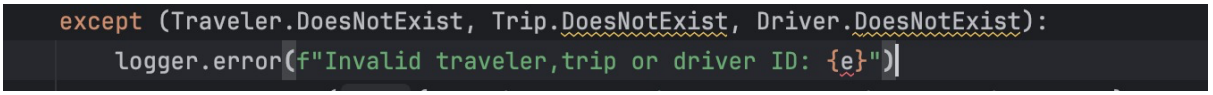

127.0.0.1:8000/api/feedback/traveler/3/
Django REST framework
blayush
Feedback List By Traveler
Feedback List By Traveler
OPTIONS
GET
GET /api/feedback/traveler/3/
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"rating": 2,
"comments": "babgjk"
}
]


127.0.0.1:8000/api/feedback/driver/1/
Django REST framework
blayush
Feedback List By Driver
Feedback List By Driver
OPTIONS
GET
GET /api/feedback/driver/1/
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"rating": 4,
"comments": "Worst driver!!!!"
}
]


127.0.0.1:8000/api/feedback/driver/3/
Django REST framework
blayush
Feedback List By Driver
Feedback List By Driver
OPTIONS
GET
GET /api/feedback/driver/3/
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"rating": 2,
"comments": "babgjk"
}
]



Exceptions and Error Handling -



C++ Implementation

Classes -

```
class Feedback;

class Trip {
public:
    Trip(const string& tripId, const string& driverName)
        : tripId(tripId), driverName(driverName) {}

    string getTripId() const {
        return tripId;
    }

    string getDriverName() const {
        return driverName;
    }

    void addFeedback(const Feedback& feedback);

    const vector<Feedback>& getFeedbacks() const {
        return feedbacks;
    }

private:
    string tripId;
    string driverName;
    vector<Feedback> feedbacks;
};

class Feedback {
public:
    Feedback(const string& travelerUsername, const string& driverName, const string& comments, const string& tripId)
        : travelerUsername(travelerUsername), driverName(driverName), comments(comments), tripId(tripId) {}

    string getTravelerUsername() const {
        return travelerUsername;
    }

    string getDriverName() const {
        return driverName;
    }

    string getComments() const {
        return comments;
    }

private:
    string travelerUsername;
    string driverName;
    string comments;
    string tripId;
};
```

Inheritance

```
class Traveler : public User {
public:
    Traveler(const string& username, const string& password)
        : User(username, password) {}

    void submitFeedback(Trip& trip, const string& feedbackText);
};

class TransportManager : public User {
public:
    TransportManager(const string& username, const string& password)
        : User(username, password) {}

    void viewAllFeedback(const vector<Trip>& trips) const;
    void viewTripFeedback(const Trip& trip) const;
    void viewAggregatedFeedback(const vector<Trip>& trips) const;
    void evaluateDriverFeedback(const vector<Trip>& trips, const string& driverName) const;
};
```

Authentication

```
class AuthenticationManager {
public:
    void addTraveler(const Traveler& traveler);
    void addTransportManager(const TransportManager& manager);
    bool authenticateUser(const string& username, const string& password);

    string generateUniqueTripId() const;

private:
    unordered_map<string, Traveler> travelers;
    unordered_map<string, TransportManager> transportManagers;
};
```

```
170
171 bool AuthenticationManager::authenticateUser(const string& username, const string& password) {
172     auto travelerIt = travelers.find(username);
173     if (travelerIt != travelers.end()) {
174         return travelerIt->second.authenticate(password);
175     }
176
177     auto managerIt = transportManagers.find(username);
178     if (managerIt != transportManagers.end()) {
179         return managerIt->second.authenticate(password);
180     }
181
182     return false;
183 }
```

Logging

```
class SystemMonitor {
public:
    static void logEvent(const string& event) {
        cout << "Event: " << event << endl;
    }
};
```

Driver Code

```cpp
int main() {
    AuthenticationManager authManager;

    // Add Traveler and TransportManager to the system
    Traveler traveler("john_traveler", "travel123");
    TransportManager transportManager("admin_transport", "adminPass");

    authManager.addTraveler(traveler);
    authManager.addTransportManager(transportManager);
    // Authenticate users
    if (authManager.authenticateUser("john_traveler", "travel123")) {
        cout << "Traveler authenticated successfully.\n";
    } else {
        cout << "Traveler authentication failed.\n";
    }

    if (authManager.authenticateUser("admin_transport", "adminPass")) {
        cout << "Transport Manager authenticated successfully.\n";
    } else {
        cout << "Transport Manager authentication failed.\n";
    }

    // Create a sample trip
    string tripId = authManager.generateUniqueTripId();
    Trip trip(tripId, "SampleDriver");

    // Traveler submits feedback
    traveler.submitFeedback(trip, "Great trip, awesome driver!");

    // Transport Manager reviews feedback
    vector<Trip> trips{trip};
    transportManager.viewAllFeedback(trips);
    transportManager.viewTripFeedback(trip);
    transportManager.viewAggregatedFeedback(trips);
    transportManager.evaluateDriverFeedback(trips, "SampleDriver");

    return 0;
}
```

Output

```
Viewing all feedback:
Trip ID: T0, Feedback: Great trip, awesome driver!
Viewing feedback for trip T0:
Feedback: Great trip, awesome driver!
Viewing aggregated feedback for all trips:
Driver: SampleDriver, Feedback: Great trip, awesome driver!
Evaluating feedback for driver SampleDriver:
Trip ID: T0, Feedback: Great trip, awesome driver!
```