

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

```
2  from django.db import models
3
4
5      9 usages
6  class Driver(models.Model):
7      name = models.CharField(max_length=255)
8      age = models.IntegerField()
9
10     10 usages
11  class Traveler(models.Model):
12      name = models.CharField(max_length=255)
13      trips = models.ManyToManyField(to='Trip', related_name='travelers')
14
15      8 usages
16  class Trip(models.Model):
17      driver = models.ForeignKey(Driver, on_delete=models.CASCADE, related_name='trips')
```

```
18      13 usages
19  class Feedback(models.Model):
20      driver = models.ForeignKey(Driver, on_delete=models.CASCADE)
21      comments = models.TextField(blank=True)
22      rating = models.PositiveIntegerField(default=0, blank=True, null=True,
23                                          choices=[(i, i) for i in range(1, 6)])
24      trip = models.ForeignKey(Trip, on_delete=models.CASCADE, related_name='feedback')
25      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)
```

```
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)
```

```
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 -

The feedback "Feedback object (3)" was added successfully.

Select feedback to change

Action: ----- Go 0 of 3 selected

ID	TRAVELER	TRIP	DRIVER	RATING	COMMENTS
3	Traveler object (3)	Trip object (2)	Driver object (3)	2	babgik
2	Traveler object (1)	Trip object (2)	Driver object (4)	5	Good!!!!
1	Traveler object (1)	Trip object (1)	Driver object (1)	4	Worst driver!!!!

3 feedbacks

Start typing to filter...

Select driver to change

Action: ----- Go 0 of 4 selected

ID	NAME
4	Kalpraj
3	Suresh
2	Rakesh
1	Ramesh

4 drivers

Start typing to filter...

Select trip to change

Action: ----- Go 0 of 4 selected

ID	DRIVER
4	Driver object (3)
3	Driver object (3)
2	Driver object (2)
1	Driver object (1)

4 trips

Start typing to filter...

Select traveler to change

Action: ----- Go 0 of 3 selected

ID	NAME
3	Vikas
2	Shivansh
1	Ayush Mishra

3 travelers

Endpoints -

Traveler List Create

GET /api/travelers/

OPTIONS GET

```
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/ blayush

Django REST framework

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

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

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

OPTIONS GET

```
GET /api/feedback/driver/3/
```

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[  
    {  
        "rating": 2,  
        "comments": "babgjk"  
    }  
]
```

The screenshot shows a browser window with the URL `127.0.0.1:8000/api/feedback/trip/1/`. The page title is "Feedback List By Trip". There are "OPTIONS" and "GET" buttons. The response is a JSON object:

```
[{"rating": 4, "comments": "Worst driver!!!!"}]
```

Exceptions and Error Handling -

```
except (Traveler.DoesNotExist, Trip.DoesNotExist, Driver.DoesNotExist):
    logger.error(f"Invalid traveler, trip or driver ID: {e}")
```

C++ Implementation

Classes -

```

4 class Feedback;
5
6 class Trip {
7 public:
8     Trip(const string& tripId, const string& driverName)
9         : tripId(tripId), driverName(driverName) {}
10
11    string getTripId() const {
12        return tripId;
13    }
14
15    string getDriverName() const {
16        return driverName;
17    }
18
19    void addFeedback(const Feedback& feedback);
20
21    const vector<Feedback>& getFeedbacks() const {
22        return feedbacks;
23    }
24
25 private:
26     string tripId;
27     string driverName;
28     vector<Feedback> feedbacks;
29 };
30
31 class Feedback {
32 public:
33     Feedback(const string& travelerUsername, const string& driverName, const string& comments, const string& tripId)
34         : travelerUsername(travelerUsername), driverName(driverName), comments(comments), tripId(tripId) {}
35
36     string getTravelerUsername() const {
37         return travelerUsername;
38     }
39
40     string getDriverName() const {
41         return driverName;
42     }
43
44     string getComments() const {
45         return comments;
46     }
47
48 private:
49     string travelerUsername;
50     string driverName;
51     string comments;
52     string tripId;
53 };

```

Inheritance

```

83 class Traveler : public User {
84 public:
85     Traveler(const string& username, const string& password)
86         : User(username, password) {}
87
88     void submitFeedback(Trip& trip, const string& feedbackText);
89 };
90
91 class TransportManager : public User {
92 public:
93     TransportManager(const string& username, const string& password)
94         : User(username, password) {}
95
96     void viewAllFeedback(const vector<Trip>& trips) const;
97     void viewTripFeedback(const Trip& trip) const;
98     void viewAggregatedFeedback(const vector<Trip>& trips) const;
99     void evaluateDriverFeedback(const vector<Trip>& trips, const string& driverName) const;
100 };
101

```

Authentication

```
165     }
166 
167     class AuthenticationManager {
168     public:
169         void addTraveler(const Traveler& traveler);
170         void addTransportManager(const TransportManager& manager);
171         bool authenticateUser(const string& username, const string& password);
172 
173         string generateUniqueTripId() const;
174 
175     private:
176         unordered_map<string, Traveler> travelers;
177         unordered_map<string, TransportManager> transportManagers;
178     };
179 }
```

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

Logging

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

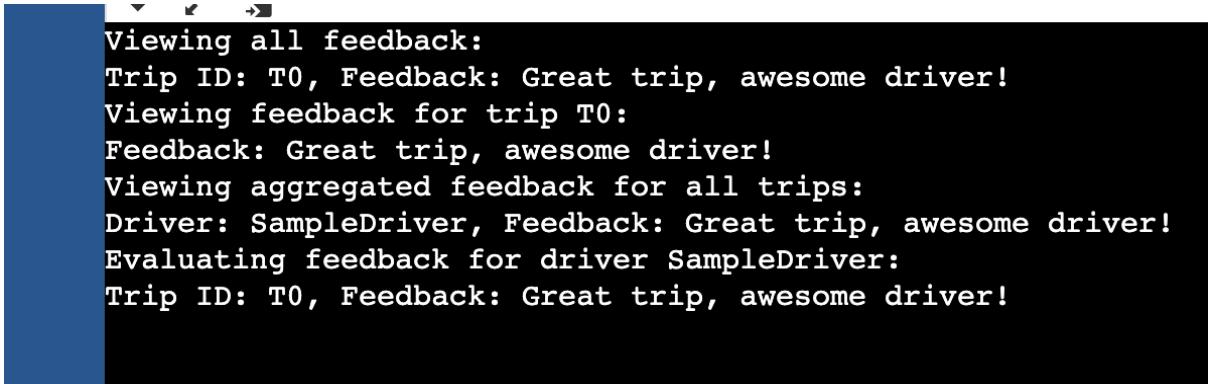
Driver Code

```

189 int main() {
190     AuthenticationManager authManager;
191
192     // Add Traveler and TransportManager to the system
193     Traveler traveler("john_traveler", "travel123");
194     TransportManager transportManager("admin_transport", "adminPass");
195
196     authManager.addTraveler(traveler);
197     authManager.addTransportManager(transportManager);
198     // Authenticate users
199     if (authManager.authenticateUser("john_traveler", "travel123")) {
200         cout << "Traveler authenticated successfully.\n";
201     } else {
202         cout << "Traveler authentication failed.\n";
203     }
204
205     if (authManager.authenticateUser("admin_transport", "adminPass")) {
206         cout << "Transport Manager authenticated successfully.\n";
207     } else {
208         cout << "Transport Manager authentication failed.\n";
209     }
210
211     // Create a sample trip
212     string tripId = authManager.generateUniqueTripId();
213     Trip trip(tripId, "SampleDriver");
214
215     // Traveler submits feedback
216     traveler.submitFeedback(trip, "Great trip, awesome driver!");
217
218     // Transport Manager reviews feedback
219     vector<Trip> trips{trip};
220     transportManager.viewAllFeedback(trips);
221     transportManager.viewTripFeedback(trip);
222     transportManager.viewAggregatedFeedback(trips);
223     transportManager.evaluateDriverFeedback(trips, "SampleDriver");
224
225
226     return 0;
227 }
228

```

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!

```