Advanced Python Programming (APP)

Course Code (CC) – CSI – 3007

Laboratory (Lab) Class Live Day To Day Assessment Assignment

Mobile App Development - Flutter

Name :- Dharshan Raj P A

Register Number :- 22MIC0073

Slot :- A2 + L7 + L8 + L27 + L28

Faculty:-Dr. Prof. Sharmila Banu K

Semester:- Fall Semester (2024-2025)

1)Weather App:-

Projects-J_Component\Flutter\lib\models\city_model.dart

```dart
class City {
  final String name;
  final double? latitude;
  final double? longitude;

  City({
    required this.name,
    this.latitude,
    this.longitude,
  });

  Map<String, dynamic> toJson() {
    return {
      'name': name,
      'latitude': latitude,
      'longitude': longitude,
    };
  }

  factory City.fromJson(Map<String, dynamic> json) {
    return City(
      name: json['name'] as String,
      latitude: json['latitude'] as double?,
      longitude: json['longitude'] as double?,
    );
  }
```

```
  @override
  bool operator ==(Object other) {
    if (identical(this, other)) return true;
    return other is City && other.name.toLowerCase() == name.toLowerCase();
  }

  @override
  int get hashCode => name.toLowerCase().hashCode;
}
```

Projects-J_Component\Flutter\lib\models\forecast_model.dart

```
class Forecast {
  final DateTime date;
  final String condition;
  final int highTemperature;
  final int lowTemperature;
  final int humidity;
  final double windSpeed;

  Forecast({
    required this.date,
    required this.condition,
    required this.highTemperature,
    required this.lowTemperature,
    required this.humidity,
    required this.windSpeed,
  });

  Map<String, dynamic> toJson() {
    return {
      'date': date.toIso8601String(),
      'condition': condition,
      'highTemperature': highTemperature,
      'lowTemperature': lowTemperature,
      'humidity': humidity,
      'windSpeed': windSpeed,
    };
  }

  factory Forecast.fromJson(Map<String, dynamic> json) {
    return Forecast(
      date: DateTime.parse(json['date'] as String),
      condition: json['condition'] as String,
      highTemperature: json['highTemperature'] as int,
```

```
      lowTemperature: json['lowTemperature'] as int,
      humidity: json['humidity'] as int,
      windSpeed: (json['windSpeed'] as num).toDouble(),
    );
  }
}
```

Projects-J_Component\Flutter\lib\models\weather_model.dart

```
class Weather {
  final String cityName;
  final String condition;
  final int temperature;
  final int humidity;
  final double windSpeed;

  Weather({
    required this.cityName,
    required this.condition,
    required this.temperature,
    required this.humidity,
    required this.windSpeed,
  });

  factory Weather.fromFlaskResponse(String response, String cityName) {
    final lines = response.split('\n');
    String condition = 'Unknown';
    int temperature = 0;
    int humidity = 0;
    double windSpeed = 0.0;

    for (var line in lines) {
      if (line.contains('Condition:')) {
        condition = line.split('Condition:')[1].trim();
      } else if (line.contains('Temperature:')) {
        final tempStr = line.split('Temperature:')[1].trim().replaceAll('°C',
'');
        temperature = int.tryParse(tempStr) ?? 0;
      } else if (line.contains('Humidity:')) {
        final humStr = line.split('Humidity:')[1].trim().replaceAll('%', '');
        humidity = int.tryParse(humStr) ?? 0;
      } else if (line.contains('Wind Speed:')) {
        final windStr = line.split('Wind Speed:')[1].trim().replaceAll('
km/h', '');
        windSpeed = double.tryParse(windStr) ?? 0.0;
      }
```

```
    }

    return Weather(
      cityName: cityName,
      condition: condition,
      temperature: temperature,
      humidity: humidity,
      windSpeed: windSpeed,
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'cityName': cityName,
      'condition': condition,
      'temperature': temperature,
      'humidity': humidity,
      'windSpeed': windSpeed,
    };
  }

  factory Weather.fromJson(Map<String, dynamic> json) {
    return Weather(
      cityName: json['cityName'] as String,
      condition: json['condition'] as String,
      temperature: json['temperature'] as int,
      humidity: json['humidity'] as int,
      windSpeed: (json['windSpeed'] as num).toDouble(),
    );
  }
}
```

Projects-J_Component\Flutter\lib\screens\favorites_screen.dart

```
import 'package:flutter/material.dart';
import '../models/city_model.dart';
import '../models/weather_model.dart';
import '../services/storage_service.dart';
import '../services/weather_service.dart';
import '../widgets/weather_card.dart';
import 'weather_detail_screen.dart';

class FavoritesScreen extends StatefulWidget {
  const FavoritesScreen({super.key});

  @override
```

```dart
  State<FavoritesScreen> createState() => _FavoritesScreenState();
}

class _FavoritesScreenState extends State<FavoritesScreen> {
  final StorageService _storageService = StorageService();
  final WeatherService _weatherService = WeatherService();
  List<City> _favorites = [];
  Map<String, Weather> _weatherCache = {};
  bool _isLoading = true;

  @override
  void initState() {
    super.initState();
    _loadFavorites();
  }

  Future<void> _loadFavorites() async {
    setState(() {
      _isLoading = true;
    });

    final favorites = await _storageService.getFavorites();

    // Load weather for each favorite
    final Map<String, Weather> cache = {};
    for (var city in favorites) {
      try {
        final weather = await _weatherService.getWeather(city.name);
        cache[city.name] = weather;
      } catch (e) {
        // Skip cities with errors
      }
    }

    setState(() {
      _favorites = favorites;
      _weatherCache = cache;
      _isLoading = false;
    });
  }

  Future<void> _removeFavorite(City city) async {
    final success = await _storageService.removeFavorite(city);
    if (success) {
      await _loadFavorites();
      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Removed from favorites')),
```

```
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Favorites'),
        actions: [
          IconButton(
            icon: const Icon(Icons.refresh),
            onPressed: _loadFavorites,
            tooltip: 'Refresh',
          ),
        ],
      ),
      body: _isLoading
          ? const Center(child: CircularProgressIndicator())
          : _favorites.isEmpty
              ? const Center(
                  child: Text('No favorite cities yet.\nSearch and add cities
to favorites.'),
                )
              : RefreshIndicator(
                  onRefresh: _loadFavorites,
                  child: ListView.builder(
                    itemCount: _favorites.length,
                    itemBuilder: (context, index) {
                      final city = _favorites[index];
                      final weather = _weatherCache[city.name];

                      if (weather == null) {
                        return ListTile(
                          title: Text(city.name),
                          trailing: IconButton(
                            icon: const Icon(Icons.delete),
                            onPressed: () => _removeFavorite(city),
                          ),
                          subtitle: const Text('Weather data unavailable'),
                        );
                      }

                      return Dismissible(
                        key: Key(city.name),
                        direction: DismissDirection.endToStart,
                        background: Container(
```

```
                        alignment: Alignment.centerRight,
                        color: Colors.red,
                        child: const Padding(
                          padding: EdgeInsets.only(right: 16.0),
                          child: Icon(Icons.delete, color: Colors.white),
                        ),
                      ),
                      onDismissed: (direction) => _removeFavorite(city),
                      child: WeatherCard(
                        weather: weather,
                        onTap: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                              builder: (context) => WeatherDetailScreen(
                                weather: weather,
                              ),
                            ),
                          ).then((_) => _loadFavorites());
                        },
                      ),
                    );
                  },
                ),
              ),
    );
  }
}
```

Projects-J_Component\Flutter\lib\screens\forecast_screen.dart

```
import 'package:flutter/material.dart';
import '../models/forecast_model.dart';
import '../services/weather_service.dart';
import '../widgets/forecast_item.dart';

class ForecastScreen extends StatefulWidget {
  final String cityName;

  const ForecastScreen({
    super.key,
    required this.cityName,
  });

  @override
  State<ForecastScreen> createState() => _ForecastScreenState();
```

```dart
}

class _ForecastScreenState extends State<ForecastScreen> {
  final WeatherService _weatherService = WeatherService();
  List<Forecast> _forecast = [];
  bool _isLoading = true;

  @override
  void initState() {
    super.initState();
    _loadForecast();
  }

  void _loadForecast() {
    setState(() {
      _isLoading = true;
    });

    final forecast = _weatherService.getForecast(widget.cityName);

    setState(() {
      _forecast = forecast;
      _isLoading = false;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('5-Day Forecast - ${widget.cityName}'),
      ),
      body: _isLoading
          ? const Center(child: CircularProgressIndicator())
          : _forecast.isEmpty
              ? const Center(child: Text('No forecast data available'))
              : ListView.builder(
                  itemCount: _forecast.length,
                  itemBuilder: (context, index) {
                    return ForecastItem(forecast: _forecast[index]);
                  },
                ),
    );
  }
}
```

Projects-J_Component\Flutter\lib\screens\home_screen.dart

```dart
import 'package:flutter/material.dart';
import '../models/weather_model.dart';
import '../services/weather_service.dart';
import '../services/location_service.dart';
import '../widgets/weather_card.dart';
import 'search_screen.dart';
import 'favorites_screen.dart';
import 'forecast_screen.dart';
import 'weather_detail_screen.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final WeatherService _weatherService = WeatherService();
  final LocationService _locationService = LocationService();
  Weather? _currentWeather;
  bool _isLoading = false;
  String? _errorMessage;

  @override
  void initState() {
    super.initState();
    _loadWeatherForCurrentLocation();
  }

  Future<void> _loadWeatherForCurrentLocation() async {
    setState(() {
      _isLoading = true;
      _errorMessage = null;
    });

    try {
      final cityName = await _locationService.getCurrentCityName();
      if (cityName != null) {
        final weather = await _weatherService.getWeather(cityName);
        setState(() {
          _currentWeather = weather;
          _isLoading = false;
        });
      } else {
        setState(() {
          _errorMessage = 'Unable to get location. Please search for a city.';
          _isLoading = false;
```

```
        });
      }
    } catch (e) {
      setState(() {
        _errorMessage = 'Error loading weather: ${e.toString()}';
        _isLoading = false;
      });
    }
  }

  Future<void> _refreshWeather() async {
    await _loadWeatherForCurrentLocation();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Weather App'),
        actions: [
          IconButton(
            icon: const Icon(Icons.search),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const SearchScreen()),
              ).then((_) => _refreshWeather());
            },
          ),
          IconButton(
            icon: const Icon(Icons.favorite),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const
FavoritesScreen()),
              ).then((_) => _refreshWeather());
            },
          ),
        ],
      ),
      body: RefreshIndicator(
        onRefresh: _refreshWeather,
        child: _isLoading
            ? const Center(child: CircularProgressIndicator())
            : _errorMessage != null
                ? Center(
                    child: Column(
```

```
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Text(
                        _errorMessage!,
                        textAlign: TextAlign.center,
                        style: const TextStyle(color: Colors.red),
                      ),
                      const SizedBox(height: 16),
                      ElevatedButton(
                        onPressed: _loadWeatherForCurrentLocation,
                        child: const Text('Retry'),
                      ),
                    ],
                  ),
                )
              : _currentWeather == null
                  ? const Center(
                      child: Text('No weather data available'),
                    )
                  : SingleChildScrollView(
                      physics: const AlwaysScrollableScrollPhysics(),
                      child: Column(
                        children: [
                          WeatherCard(
                            weather: _currentWeather!,
                            onTap: () {
                              Navigator.push(
                                context,
                                MaterialPageRoute(
                                  builder: (context) => WeatherDetailScreen(
                                    weather: _currentWeather!,
                                  ),
                                ),
                              );
                            },
                          ),
                          Padding(
                            padding: const EdgeInsets.all(16.0),
                            child: ElevatedButton.icon(
                              onPressed: () {
                                Navigator.push(
                                  context,
                                  MaterialPageRoute(
                                    builder: (context) => ForecastScreen(
                                      cityName: _currentWeather!.cityName,
                                    ),
                                  ),
                                );
```

```
                                },
                                icon: const Icon(Icons.calendar_today),
                                label: const Text('View 5-Day Forecast'),
                                style: ElevatedButton.styleFrom(
                                  padding: const EdgeInsets.symmetric(
                                    horizontal: 24,
                                    vertical: 12,
                                  ),
                                ),
                              ),
                            ),
                          ),
                        ],
                      ),
                    ),
                  ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _loadWeatherForCurrentLocation,
        tooltip: 'Refresh Location',
        child: const Icon(Icons.my_location),
      ),
    );
  }
}
```

Projects-J_Component\Flutter\lib\screens\search_screen.dart

```
import 'package:flutter/material.dart';
import '../models/weather_model.dart';
import '../models/city_model.dart';
import '../services/weather_service.dart';
import '../services/storage_service.dart';
import '../widgets/weather_card.dart';
import 'weather_detail_screen.dart';

class SearchScreen extends StatefulWidget {
  const SearchScreen({super.key});

  @override
  State<SearchScreen> createState() => _SearchScreenState();
}

class _SearchScreenState extends State<SearchScreen> {
  final WeatherService _weatherService = WeatherService();
  final StorageService _storageService = StorageService();
  final TextEditingController _searchController = TextEditingController();
  Weather? _weather;
  bool _isLoading = false;
  String? _errorMessage;
```

```dart
bool _isFavorite = false;

@override
void dispose() {
  _searchController.dispose();
  super.dispose();
}

Future<void> _searchWeather(String cityName) async {
  if (cityName.trim().isEmpty) {
    setState(() {
      _errorMessage = 'Please enter a city name';
    });
    return;
  }

  setState(() {
    _isLoading = true;
    _errorMessage = null;
    _weather = null;
  });

  try {
    final weather = await _weatherService.getWeather(cityName.trim());
    final isFav = await _storageService.isFavorite(cityName.trim());

    setState(() {
      _weather = weather;
      _isFavorite = isFav;
      _isLoading = false;
    });
  } catch (e) {
    setState(() {
      _errorMessage = 'Error loading weather: ${e.toString()}';
      _isLoading = false;
    });
  }
}

Future<void> _toggleFavorite() async {
  if (_weather == null) return;

  final city = City(
    name: _weather!.cityName,
  );

  bool success;
  if (_isFavorite) {
```

```
      success = await _storageService.removeFavorite(city);
    } else {
      success = await _storageService.addFavorite(city);
    }

    if (success) {
      setState(() {
        _isFavorite = !_isFavorite;
      });

      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(
              _isFavorite
                  ? 'Added to favorites'
                  : 'Removed from favorites',
            ),
          ),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Search City'),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: Row(
              children: [
                Expanded(
                  child: TextField(
                    controller: _searchController,
                    decoration: const InputDecoration(
                      labelText: 'Enter city name',
                      border: OutlineInputBorder(),
                      prefixIcon: Icon(Icons.search),
                    ),
                    onSubmitted: _searchWeather,
                  ),
                ),
                const SizedBox(width: 8),
```

```
                ElevatedButton(
                  onPressed: () => _searchWeather(_searchController.text),
                  child: const Text('Search'),
                ),
              ],
            ),
          ),
          if (_isLoading)
            const Expanded(
              child: Center(child: CircularProgressIndicator()),
            )
          else if (_errorMessage != null)
            Expanded(
              child: Center(
                child: Text(
                  _errorMessage!,
                  style: const TextStyle(color: Colors.red),
                  textAlign: TextAlign.center,
                ),
              ),
            )
          else if (_weather != null)
            Expanded(
              child: SingleChildScrollView(
                child: Column(
                  children: [
                    WeatherCard(
                      weather: _weather!,
                      onTap: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) => WeatherDetailScreen(
                              weather: _weather!,
                            ),
                          ),
                        );
                      },
                    ),
                    Padding(
                      padding: const EdgeInsets.symmetric(horizontal: 16.0),
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          IconButton(
                            icon: Icon(
                              _isFavorite ? Icons.favorite :
Icons.favorite_border,
```

```
                            color: _isFavorite ? Colors.red : null,
                          ),
                          onPressed: _toggleFavorite,
                          tooltip: _isFavorite
                              ? 'Remove from favorites'
                              : 'Add to favorites',
                        ),
                        const SizedBox(width: 8),
                        Text(
                          _isFavorite
                              ? 'In Favorites'
                              : 'Add to Favorites',
                          style: const TextStyle(fontSize: 16),
                        ),
                      ],
                    ),
                  ),
                ],
              ),
            ),
          )
        else
          const Expanded(
            child: Center(
              child: Text('Search for a city to see weather information'),
            ),
          ),
      ],
    ),
  );
 }
}
```

Projects-J_Component\Flutter\lib\screens\weather_detail_screen.dart

```
import 'package:flutter/material.dart';
import '../models/weather_model.dart';
import '../services/weather_service.dart';
import '../models/forecast_model.dart';
import '../widgets/forecast_item.dart';
import 'forecast_screen.dart';

class WeatherDetailScreen extends StatelessWidget {
  final Weather weather;

  const WeatherDetailScreen({
```

```
    super.key,
    required this.weather,
  });

  String _getWeatherIcon(String condition) {
    switch (condition.toLowerCase()) {
      case 'sunny':
        return '☀';
      case 'cloudy':
        return '☁';
      case 'rainy':
        return '🌧';
      case 'windy':
        return '💨';
      case 'stormy':
        return '⛈';
      case 'clear night':
        return '🌙';
      default:
        return '🌤';
    }
  }

  Color _getWeatherColor(String condition) {
    switch (condition.toLowerCase()) {
      case 'sunny':
        return Colors.orange;
      case 'cloudy':
        return Colors.grey;
      case 'rainy':
        return Colors.blue;
      case 'windy':
        return Colors.blueGrey;
      case 'stormy':
        return Colors.purple;
      case 'clear night':
        return Colors.indigo;
      default:
        return Colors.blue;
    }
  }

  @override
  Widget build(BuildContext context) {
    final weatherService = WeatherService();
    final forecast = weatherService.getForecast(weather.cityName);

    return Scaffold(
```

```
      appBar: AppBar(
        title: Text(weather.cityName),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            // Main weather display
            Container(
              width: double.infinity,
              padding: const EdgeInsets.all(32.0),
              decoration: BoxDecoration(
                gradient: LinearGradient(
                  begin: Alignment.topCenter,
                  end: Alignment.bottomCenter,
                  colors: [
                    _getWeatherColor(weather.condition),
                    _getWeatherColor(weather.condition).withOpacity(0.7),
                  ],
                ),
              ),
              child: Column(
                children: [
                  Text(
                    _getWeatherIcon(weather.condition),
                    style: const TextStyle(fontSize: 80),
                  ),
                  const SizedBox(height: 16),
                  Text(
                    weather.condition,
                    style: const TextStyle(
                      fontSize: 32,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                    ),
                  ),
                  const SizedBox(height: 8),
                  Text(
                    '${weather.temperature}°C',
                    style: const TextStyle(
                      fontSize: 64,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                    ),
                  ),
                ],
              ),
            ),
            // Weather details
```

```
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const Text(
                    'Weather Details',
                    style: TextStyle(
                      fontSize: 24,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  const SizedBox(height: 16),
                  _buildDetailCard(
                    icon: Icons.water_drop,
                    label: 'Humidity',
                    value: '${weather.humidity}%',
                  ),
                  const SizedBox(height: 8),
                  _buildDetailCard(
                    icon: Icons.air,
                    label: 'Wind Speed',
                    value: '${weather.windSpeed} km/h',
                  ),
                ],
              ),
            ),
            // Forecast preview
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: [
                      const Text(
                        '5-Day Forecast',
                        style: TextStyle(
                          fontSize: 24,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      TextButton(
                        onPressed: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
```

```
                            builder: (context) => ForecastScreen(
                              cityName: weather.cityName,
                            ),
                          ),
                        );
                      },
                      child: const Text('View All'),
                    ),
                  ],
                ),
                const SizedBox(height: 8),
                ...forecast.take(3).map((f) => ForecastItem(forecast: f)),
              ],
            ),
          ),
        ],
      ),
    ),
  );
 }

 Widget _buildDetailCard({
   required IconData icon,
   required String label,
   required String value,
 }) {
   return Card(
     child: ListTile(
       leading: Icon(icon, size: 32),
       title: Text(label),
       trailing: Text(
         value,
         style: const TextStyle(
           fontSize: 18,
           fontWeight: FontWeight.bold,
         ),
       ),
     ),
   );
 }
}
```

Projects-J_Component\Flutter\lib\services\location_service.dart

```
import 'package:geolocator/geolocator.dart';
```

```dart
class LocationService {
  Future<Position?> getCurrentPosition() async {
    bool serviceEnabled;
    LocationPermission permission;

    // Check if location services are enabled
    serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
      return null;
    }

    // Check location permissions
    permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
      permission = await Geolocator.requestPermission();
      if (permission == LocationPermission.denied) {
        return null;
      }
    }

    if (permission == LocationPermission.deniedForever) {
      return null;
    }

    // Get current position
    try {
      return await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.medium,
      );
    } catch (e) {
      return null;
    }
  }

  Future<String?> getCityNameFromLocation(Position position) async {
    try {
      // Using reverse geocoding to get city name
      // Note: This is a simplified version. In production, you might want to
use
      // a geocoding service like Google Geocoding API or OpenStreetMap
Nominatim
      final placemarks = await Geolocator.placemarkFromCoordinates(
        position.latitude,
        position.longitude,
      );

      if (placemarks.isNotEmpty) {
        final placemark = placemarks.first;
```

```
        return placemark.locality ?? placemark.subAdministrativeArea ??
'Unknown';
      }
      return null;
    } catch (e) {
      return null;
    }
  }

  Future<String?> getCurrentCityName() async {
    final position = await getCurrentPosition();
    if (position != null) {
      return await getCityNameFromLocation(position);
    }
    return null;
  }
}
```

Projects-J_Component\Flutter\lib\services\mock_weather_service.dart

```
import 'dart:math';
import '../models/weather_model.dart';
import '../models/forecast_model.dart';

class MockWeatherService {
  static final Random _random = Random();
  static final List<String> _conditions = [
    'Sunny',
    'Cloudy',
    'Rainy',
    'Windy',
    'Stormy',
    'Clear Night',
  ];

  Weather getMockWeather(String cityName) {
    return Weather(
      cityName: cityName,
      condition: _conditions[_random.nextInt(_conditions.length)],
      temperature: 20 + _random.nextInt(19), // 20-38°C
      humidity: 40 + _random.nextInt(51), // 40-90%
      windSpeed: (2.0 + _random.nextDouble() * 10.0).roundToDouble() / 10, //
2.0-12.0 km/h with 1 decimal
    );
  }
```

```
  List<Forecast> getMockForecast(String cityName) {
    final List<Forecast> forecast = [];
    final now = DateTime.now();

    for (int i = 0; i < 5; i++) {
      final date = now.add(Duration(days: i + 1));
      final baseTemp = 20 + _random.nextInt(19);
      final highTemp = baseTemp + _random.nextInt(5);
      final lowTemp = baseTemp - _random.nextInt(5);

      forecast.add(Forecast(
        date: date,
        condition: _conditions[_random.nextInt(_conditions.length)],
        highTemperature: highTemp.clamp(20, 38),
        lowTemperature: lowTemp.clamp(15, 35),
        humidity: 40 + _random.nextInt(51),
        windSpeed: (2.0 + _random.nextDouble() * 10.0).roundToDouble() / 10,
      ));
    }

    return forecast;
  }
}
```

Projects-J_Component\Flutter\lib\services\storage_service.dart

```
import 'dart:convert';
import 'package:shared_preferences/shared_preferences.dart';
import '../models/city_model.dart';

class StorageService {
  static const String _favoritesKey = 'favorite_cities';

  Future<List<City>> getFavorites() async {
    try {
      final prefs = await SharedPreferences.getInstance();
      final favoritesJson = prefs.getString(_favoritesKey);

      if (favoritesJson == null) {
        return [];
      }

      final List<dynamic> favoritesList = json.decode(favoritesJson);
      return favoritesList
          .map((json) => City.fromJson(json as Map<String, dynamic>))
          .toList();
```

```
    } catch (e) {
      return [];
    }
  }

  Future<bool> addFavorite(City city) async {
    try {
      final favorites = await getFavorites();

      // Check if city already exists
      if (favorites.any((c) => c.name.toLowerCase() ==
city.name.toLowerCase())) {
        return false;
      }

      favorites.add(city);
      return await _saveFavorites(favorites);
    } catch (e) {
      return false;
    }
  }

  Future<bool> removeFavorite(City city) async {
    try {
      final favorites = await getFavorites();
      favorites.removeWhere((c) => c.name.toLowerCase() ==
city.name.toLowerCase());
      return await _saveFavorites(favorites);
    } catch (e) {
      return false;
    }
  }

  Future<bool> isFavorite(String cityName) async {
    try {
      final favorites = await getFavorites();
      return favorites.any((c) => c.name.toLowerCase() ==
cityName.toLowerCase());
    } catch (e) {
      return false;
    }
  }

  Future<bool> _saveFavorites(List<City> favorites) async {
    try {
      final prefs = await SharedPreferences.getInstance();
      final favoritesJson = json.encode(
        favorites.map((city) => city.toJson()).toList(),
```

```
      );
      return await prefs.setString(_favoritesKey, favoritesJson);
    } catch (e) {
      return false;
    }
  }
}
```

Projects-J_Component\Flutter\lib\services\weather_api_service.dart

```
import 'dart:io';
import 'package:http/http.dart' as http;
import '../models/weather_model.dart';

class WeatherApiService {
  static const String baseUrl = 'http://127.0.0.1:5005';
  static const Duration timeout = Duration(seconds: 5);

  Future<Weather?> getWeather(String cityName) async {
    try {
      final uri = Uri.parse('$baseUrl/weather?city=$cityName');
      final response = await http
          .get(uri)
          .timeout(timeout);

      if (response.statusCode == 200) {
        return Weather.fromFlaskResponse(response.body, cityName);
      } else {
        return null;
      }
    } on SocketException {
      // Network error - Flask server not running
      return null;
    } on HttpException {
      // HTTP error
      return null;
    } catch (e) {
      // Any other error
      return null;
    }
  }
}
```

Projects-J_Component\Flutter\lib\services\weather_service.dart

```dart
import '../models/weather_model.dart';
import '../models/forecast_model.dart';
import 'weather_api_service.dart';
import 'mock_weather_service.dart';

class WeatherService {
  final WeatherApiService _apiService = WeatherApiService();
  final MockWeatherService _mockService = MockWeatherService();

  Future<Weather> getWeather(String cityName) async {
    // Try Flask API first
    final apiWeather = await _apiService.getWeather(cityName);

    if (apiWeather != null) {
      return apiWeather;
    }

    // Fallback to mock data
    return _mockService.getMockWeather(cityName);
  }

  List<Forecast> getForecast(String cityName) {
    // Flask API doesn't provide forecast, so always use mock data
    return _mockService.getMockForecast(cityName);
  }
}
```

Projects-J_Component\Flutter\lib\widgets\forecast_item.dart

```dart
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import '../models/forecast_model.dart';

class ForecastItem extends StatelessWidget {
  final Forecast forecast;

  const ForecastItem({
    super.key,
    required this.forecast,
  });

  String _getWeatherIcon(String condition) {
    switch (condition.toLowerCase()) {
      case 'sunny':
        return '☀';
      case 'cloudy':
```

```
        return '☁️';
      case 'rainy':
        return '🌧️';
      case 'windy':
        return '💨';
      case 'stormy':
        return '⛈️';
      case 'clear night':
        return '🌙';
      default:
        return '🌤️';
    }
  }

  @override
  Widget build(BuildContext context) {
    final dateFormat = DateFormat('EEEE, MMM d');
    final dayName = DateFormat('EEEE').format(forecast.date);

    return Card(
      margin: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
      child: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Row(
          children: [
            Expanded(
              flex: 2,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    dayName,
                    style: const TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  Text(
                    dateFormat.format(forecast.date),
                    style: const TextStyle(
                      fontSize: 12,
                      color: Colors.grey,
                    ),
                  ),
                ],
              ),
            ),
            Expanded(
```

```
              child: Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text(
                    _getWeatherIcon(forecast.condition),
                    style: const TextStyle(fontSize: 32),
                  ),
                  const SizedBox(width: 8),
                  Text(
                    forecast.condition,
                    style: const TextStyle(fontSize: 14),
                  ),
                ],
              ),
            ),
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.end,
                children: [
                  Text(
                    '${forecast.highTemperature}°',
                    style: const TextStyle(
                      fontSize: 18,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  Text(
                    '${forecast.lowTemperature}°',
                    style: const TextStyle(
                      fontSize: 14,
                      color: Colors.grey,
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Projects-J_Component\Flutter\lib\widgets\weather_card.dart

```
import 'package:flutter/material.dart';
```

```dart
import '../models/weather_model.dart';

class WeatherCard extends StatelessWidget {
  final Weather weather;
  final VoidCallback? onTap;

  const WeatherCard({
    super.key,
    required this.weather,
    this.onTap,
  });

  String _getWeatherIcon(String condition) {
    switch (condition.toLowerCase()) {
      case 'sunny':
        return '☀️';
      case 'cloudy':
        return '☁️';
      case 'rainy':
        return '🌧️';
      case 'windy':
        return '💨';
      case 'stormy':
        return '⛈️';
      case 'clear night':
        return '🌙';
      default:
        return '🌤️';
    }
  }

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 4,
      margin: const EdgeInsets.all(8),
      child: InkWell(
        onTap: onTap,
        borderRadius: BorderRadius.circular(12),
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Expanded(
```

```
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      weather.cityName,
                      style: const TextStyle(
                        fontSize: 24,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    const SizedBox(height: 8),
                    Row(
                      children: [
                        Text(
                          _getWeatherIcon(weather.condition),
                          style: const TextStyle(fontSize: 32),
                        ),
                        const SizedBox(width: 8),
                        Text(
                          weather.condition,
                          style: const TextStyle(
                            fontSize: 18,
                            color: Colors.grey,
                          ),
                        ),
                      ],
                    ),
                  ],
                ),
              ),
              Text(
                '${weather.temperature}°C',
                style: const TextStyle(
                  fontSize: 48,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
          const Divider(height: 24),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
              _buildInfoItem('Humidity', '${weather.humidity}%'),
              _buildInfoItem('Wind Speed', '${weather.windSpeed} km/h'),
            ],
          ),
        ],
```

```
          ),
        ),
      ),
    );
  }

  Widget _buildInfoItem(String label, String value) {
    return Column(
      children: [
        Text(
          label,
          style: const TextStyle(
            fontSize: 12,
            color: Colors.grey,
          ),
        ),
        const SizedBox(height: 4),
        Text(
          value,
          style: const TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.w600,
          ),
        ),
      ],
    );
  }
}
```

Projects-J_Component\Flutter\lib\main.dart

```
import 'package:flutter/material.dart';
import 'screens/home_screen.dart';

void main() {
  runApp(const WeatherApp());
}

class WeatherApp extends StatelessWidget {
  const WeatherApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Weather App',
      theme: ThemeData(
```

```
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ),
      home: const HomeScreen(),
      debugShowCheckedModeBanner: false,
    );
  }
}
```

Projects-J_Component\Flutter\analysis_options.yaml

```
include: package:flutter_lints/flutter.yaml

linter:
  rules:
    prefer_const_constructors: true
    prefer_const_literals_to_create_immutables: true
    avoid_print: true
```

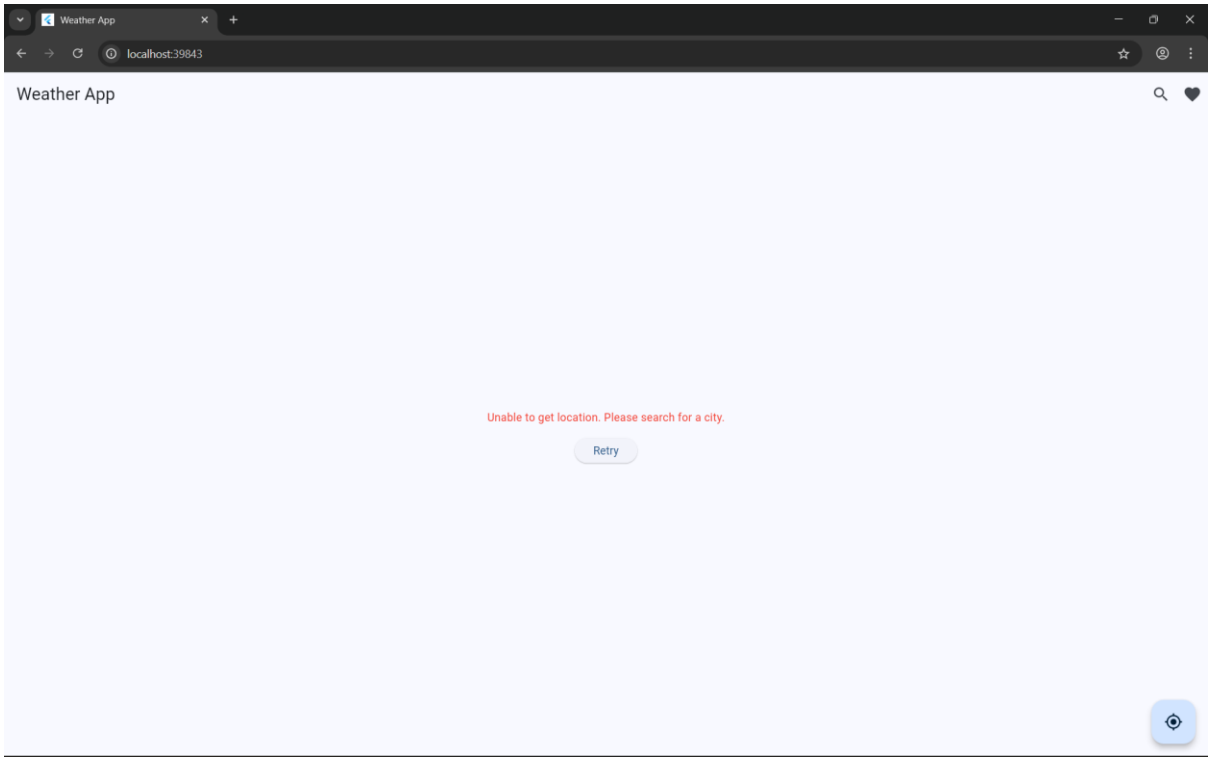
Projects-J_Component\Flutter\pubspec.yaml

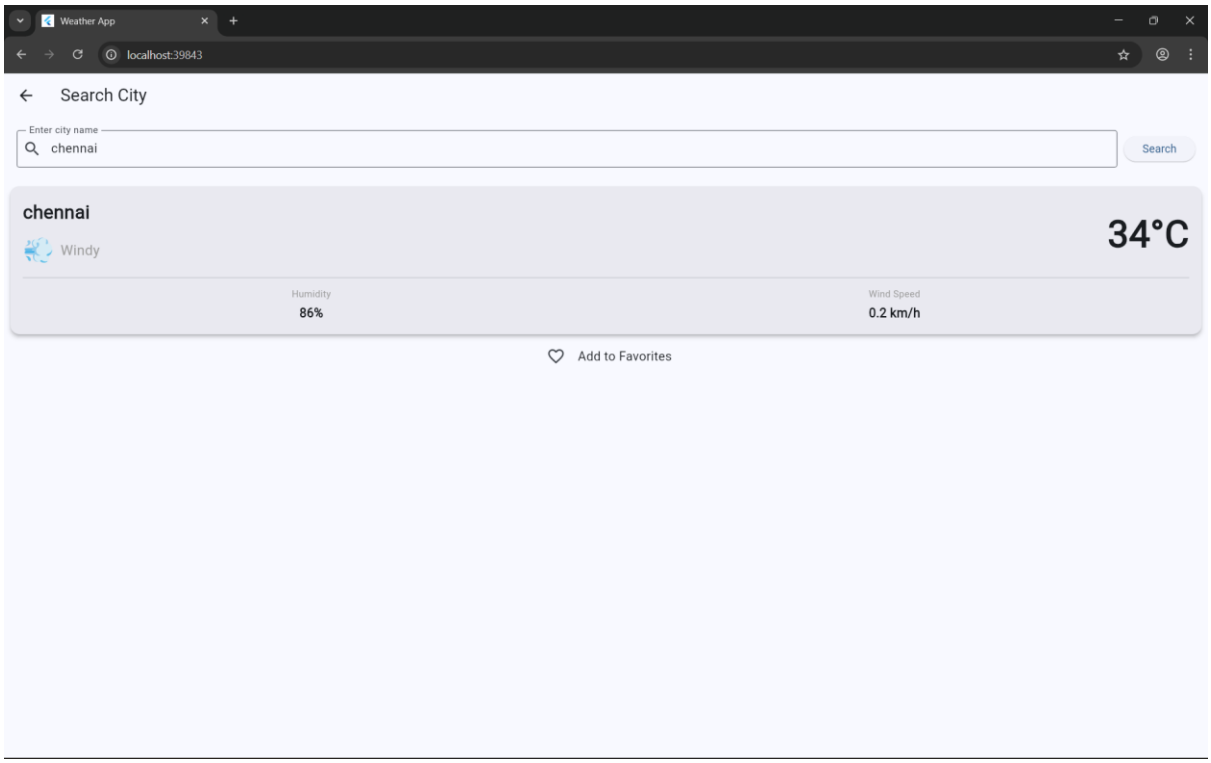
```
name: weather_app
description: A Flutter weather app prototype that integrates with Flask
weather API
publish_to: 'none'
version: 1.0.0+1

environment:
  sdk: '>=3.0.0 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  http: ^1.1.0
  geolocator: ^10.1.0
  shared_preferences: ^2.2.2
  intl: ^0.18.1
  cupertino_icons: ^1.0.6

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0

flutter:
  uses-material-design: true
```

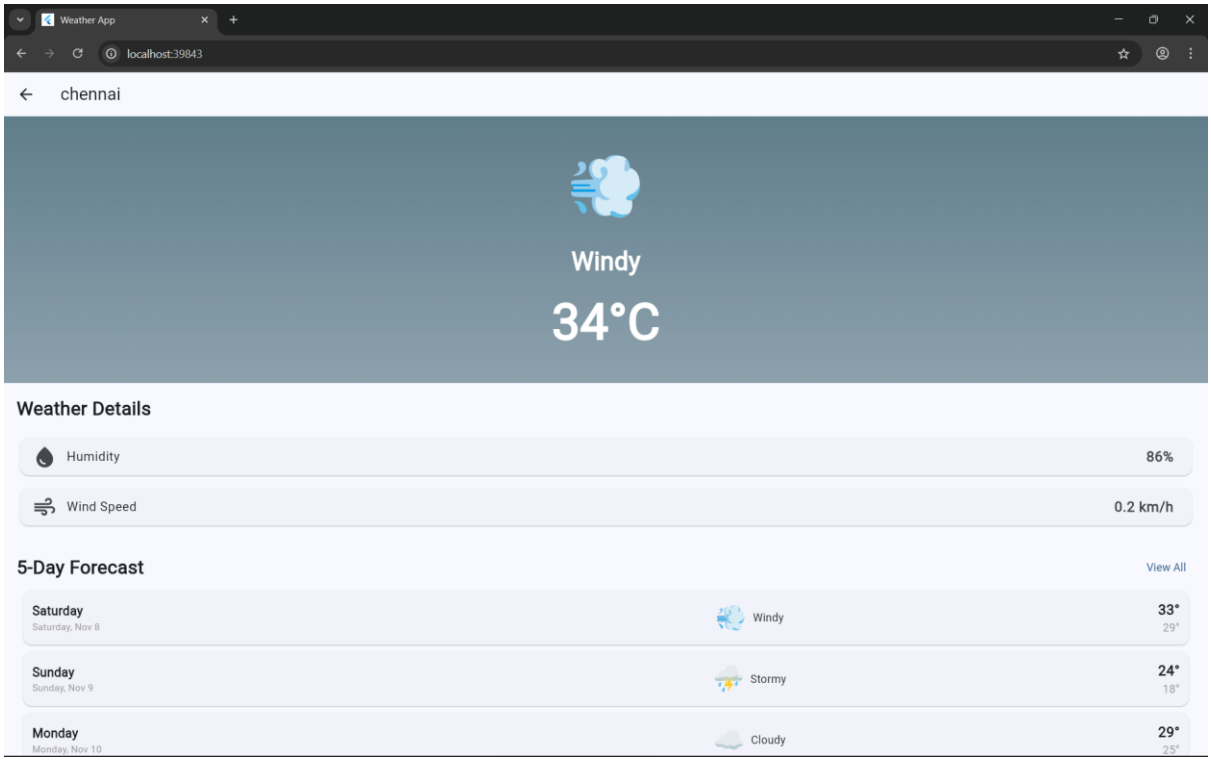
Output Screenshots Snip Snap Terminal Bash Command Prompt Shell:-

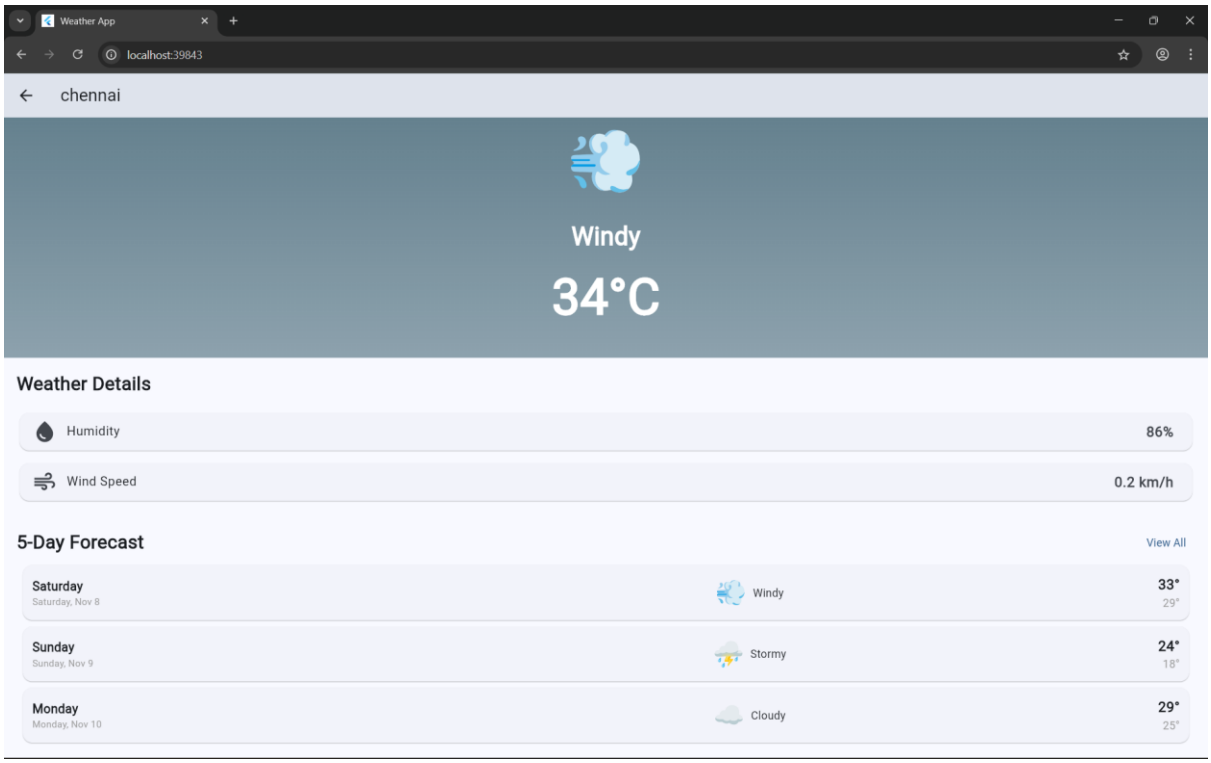
```
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter config --enable-web
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter pub get
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter create --platforms=web .
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter flutter run -d chrome
```

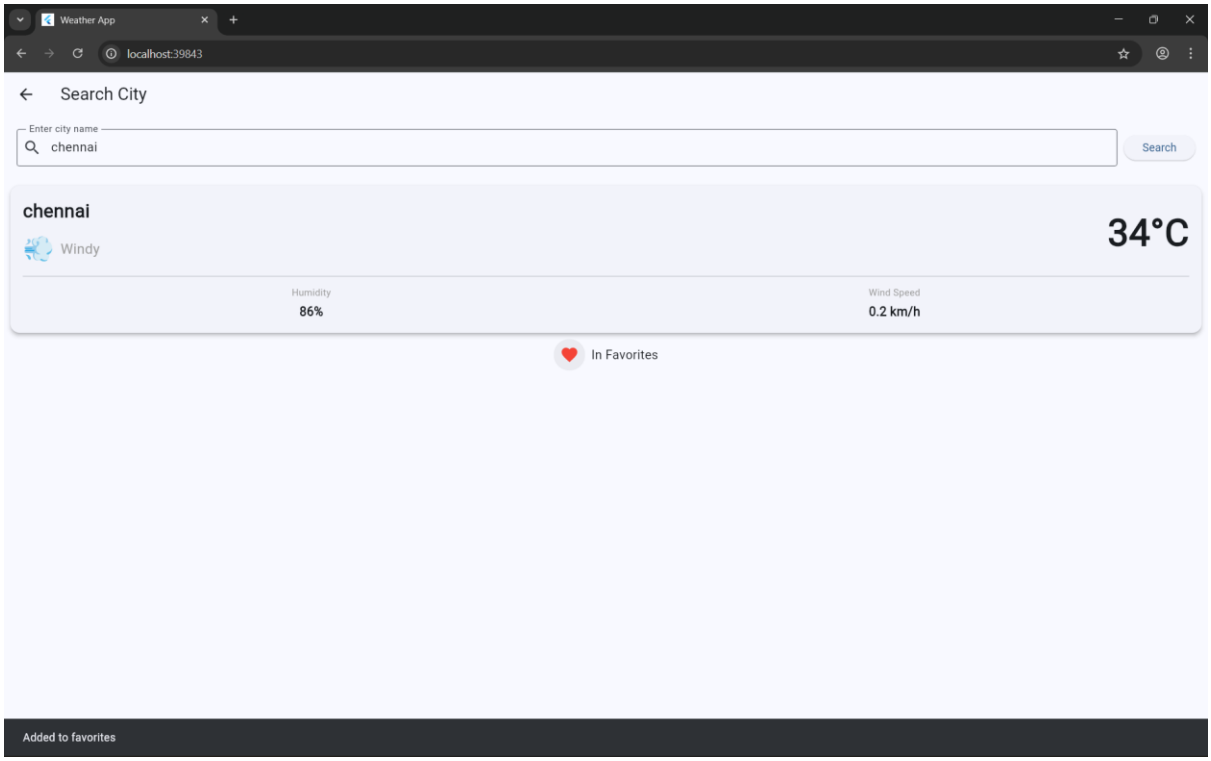

```
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter run -d windows
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter config --enable-web
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter pub get
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter create --platforms=web .
PS C:\Dharshan Raj P A\College\Laboratory\Advanced_Python_Programming_Laboratory\Projects-J_Component\Flutter> C:\Users\
dhars\OneDrive\Desktop1\flutter_windows_3.35.7-stable\flutter\bin\flutter flutter run -d chrome
```



Weather App
localhost:39843
Weather App
Unable to get location. Please search for a city.
Retry

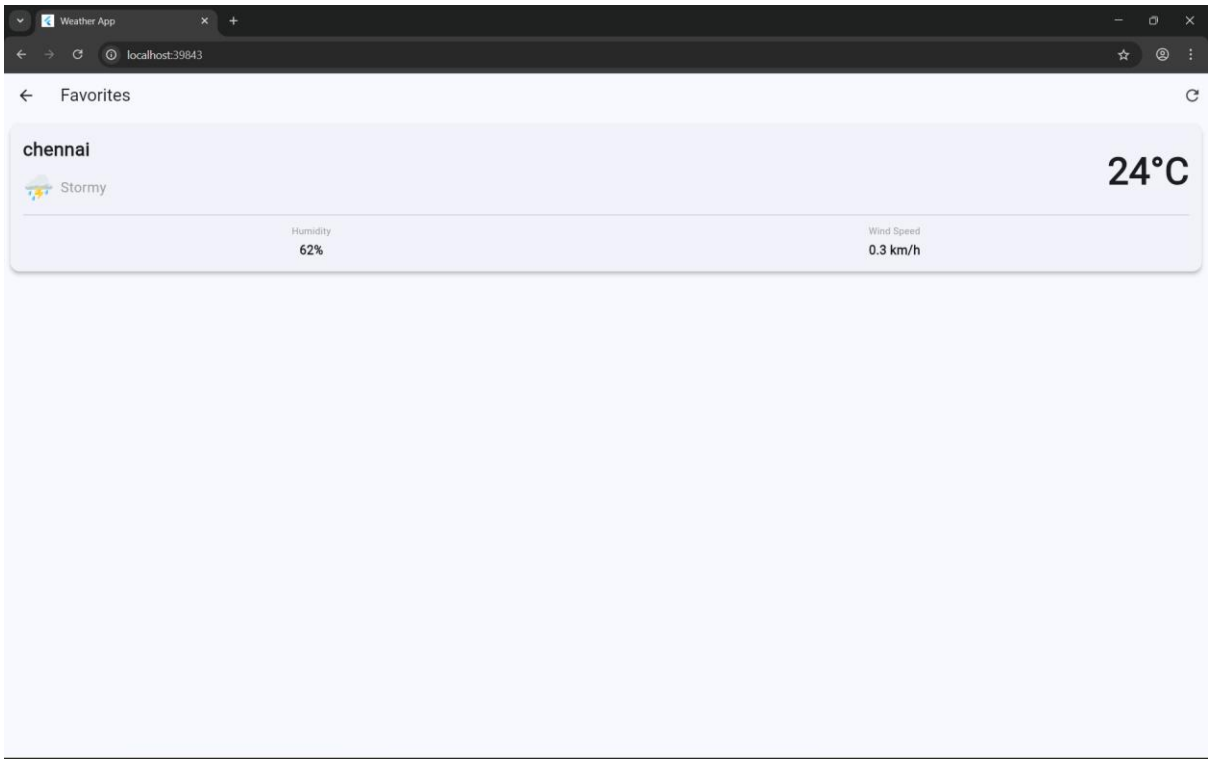

Weather App
localhost:39843
Search City
Enter city name
chennai
Search
chennai
Windy
34°C
Humidity
86%
Wind Speed
0.2 km/h
Add to Favorites


Weather App
localhost:39843
chennai
Windy
34°C
Weather Details
Humidity
86%
Wind Speed
0.2 km/h
5-Day Forecast
View All
Saturday
Saturday, Nov 8
Windy
33°
29°
Sunday
Sunday, Nov 9
Stormy
24°
18°
Monday
Monday, Nov 10
Cloudy
29°
25°


Weather App
localhost:39843
chennai
Windy
34°C
Weather Details
Humidity
86%
Wind Speed
0.2 km/h
5-Day Forecast
View All
Saturday
Saturday, Nov 8
Windy
33°
29°
Sunday
Sunday, Nov 9
Stormy
24°
18°
Monday
Monday, Nov 10
Cloudy
29°
25°


Weather App
localhost:39843
Search City
Enter city name
chennai
Search
chennai
Windy
34°C
Humidity
86%
Wind Speed
0.2 km/h
In Favorites

Added to favorites


Weather App
localhost:39843
Favorites
chennai
Stormy
24°C
Humidity
62%
Wind Speed
0.3 km/h