Certainly, here's a simplified outline of the coding steps for developing a Flutter app to display real-time parking availability data received from a Raspberry Pi:

1. **Flutter Setup**: Make sure you have Flutter and Dart installed. If not, follow the installation guide on the official Flutter website.

2. **Create a New Flutter Project**:
   ```bash
   flutter create parking_app
   cd parking_app
   ```

3. **UI Design**:
   - Open the `lib/main.dart` file and define the app's UI using Flutter widgets.

4. **Network Requests**:
   - Use the `http` package in Flutter to make HTTP requests to your Raspberry Pi server for parking data.
   - Parse the JSON data received from the server.

```dart
import 'package:http/http.dart' as http;

Future<Map<String, dynamic>> fetchParkingAvailability() async {
  final response = await http.get('http://your-raspberry-pi-ip/parking-data');

  if (response.statusCode == 200) {
    return json.decode(response.body);
  } else {
    throw Exception('Failed to load parking availability');
  }
}
```

5. **Display Data**:
   - Update your UI to display the parking availability data.

```dart
FutureBuilder<Map<String, dynamic>>(
  future: fetchParkingAvailability(),
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return CircularProgressIndicator();
    } else if (snapshot.hasError) {
      return Text('Error: ${snapshot.error}');
    } else {
      // Display parking data using Flutter widgets
      return Text('Available Spaces: ${snapshot.data['available_spaces']}');
```

```
    }
  },
)
```

6. **Auto-Refresh**:
   - Implement a mechanism to periodically refresh the parking data. You can use `Timer` or other scheduling methods.

7. **Testing and Deployment:**
   - Test your app on emulators and real devices.
   - To deploy to app stores, you'll need to follow their respective guidelines.

8. **Security:**
   - Ensure secure communication between the app and Raspberry Pi. You may need to handle authentication and encryption based on your specific requirements.

This is a simplified outline to get you started with coding. You'll need to expand on these steps and structure your code according to your specific requirements. Additionally, don't forget to add error handling, manage state, and consider the overall user experience in your app.