Gas Station Finder - Implementation Manual

Introduction

This manual provides a detailed overview of the design and implementation of the Gas Station Finder application, which allows users to search for gas stations by zip code and submit reviews for each station. The application is built using JavaFX and follows the MVC (Model-View-Controller) design pattern.

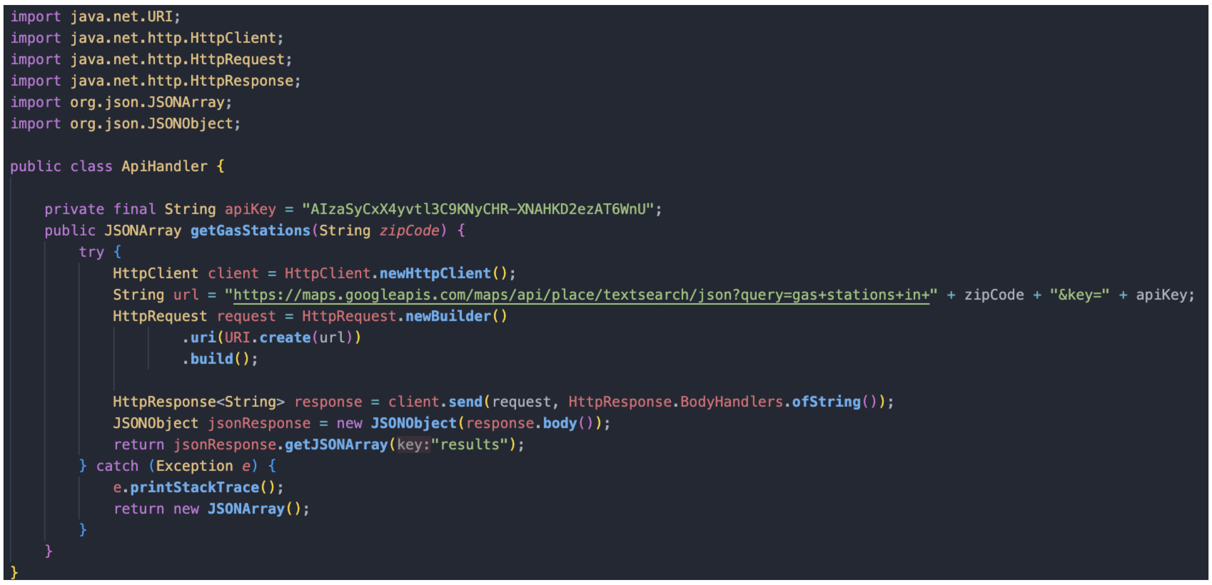
Architecture Overview

The application's architecture consists of several key components, as illustrated in the UML Class Diagram provided during the project proposal phase.

ApiHandler Class

The ApiHandler class is responsible for communicating with the Google Places API. It constructs the HTTP request, sends it, and processes the response.

Method getGasStations: This method takes a zip code as a parameter and returns an JSONArray of gas stations.

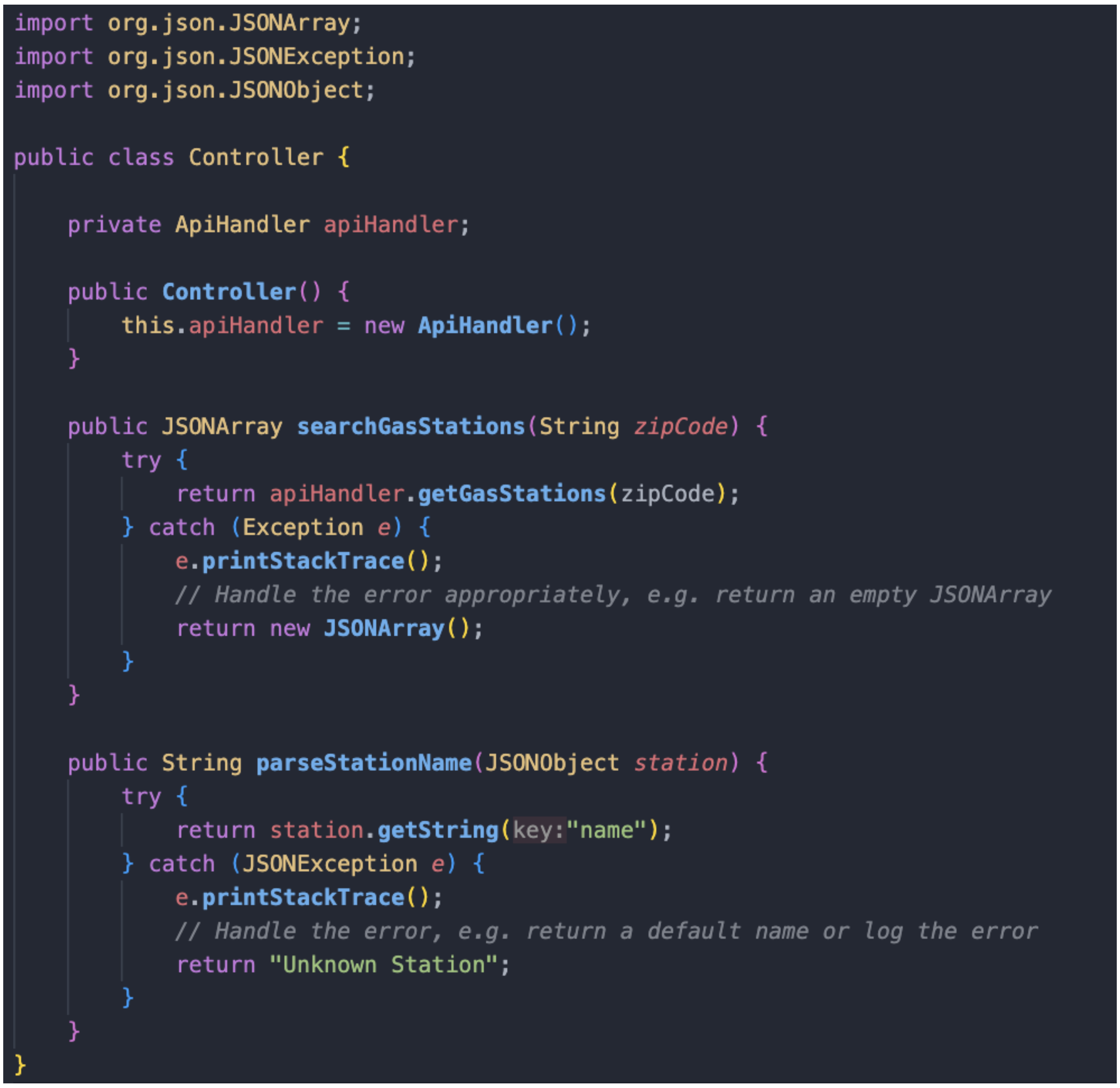


```java
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import org.json.JSONArray;
import org.json.JSONObject;

public class ApiHandler {

    private final String apiKey = "AIzaSyCxX4yvtl3C9KNyCHR-XNAHKD2ezAT6WnU";
    public JSONArray getGasStations(String zipCode) {
        try {
            HttpClient client = HttpClient.newHttpClient();
            String url = "https://maps.googleapis.com/maps/api/place/textsearch/json?query=gas+stations+in+" + zipCode + "&key=" + apiKey;
            HttpRequest request = HttpRequest.newBuilder()
                    .uri(URI.create(url))
                    .build();

            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
            JSONObject jsonResponse = new JSONObject(response.body());
            return jsonResponse.getJSONArray(key:"results");
        } catch (Exception e) {
            e.printStackTrace();
            return new JSONArray();
        }
    }
}
```

Controller Class

The Controller class serves as the intermediary between the ApiHandler and the Main class. It utilizes the ApiHandler to fetch the data and then parses it for the Main class to display.

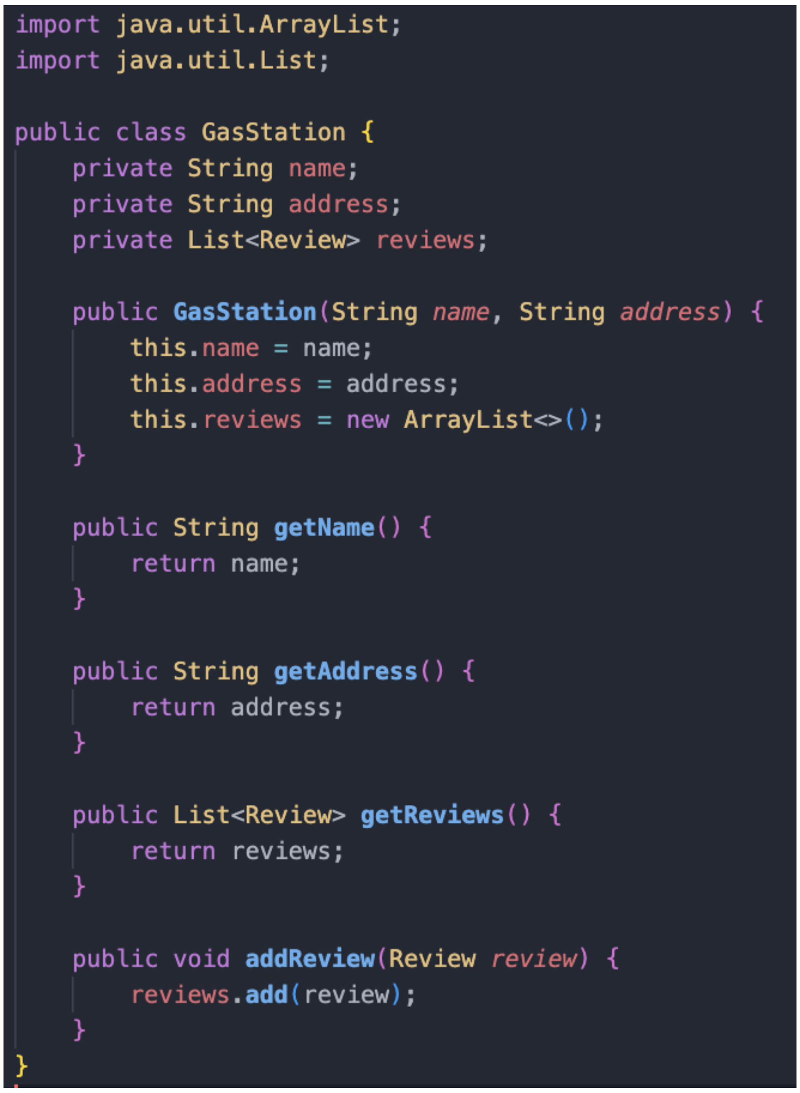
Method searchGasStations: Calls getGasStations from ApiHandler and returns the results.
Method parseStationName: Extracts the name of a gas station from the JSON object.



```java
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class Controller {

    private ApiHandler apiHandler;

    public Controller() {
        this.apiHandler = new ApiHandler();
    }

    public JSONArray searchGasStations(String zipCode) {
        try {
            return apiHandler.getGasStations(zipCode);
        } catch (Exception e) {
            e.printStackTrace();
            // Handle the error appropriately, e.g. return an empty JSONArray
            return new JSONArray();
        }
    }

    public String parseStationName(JSONObject station) {
        try {
            return station.getString(key:"name");
        } catch (JSONException e) {
            e.printStackTrace();
            // Handle the error, e.g. return a default name or log the error
            return "Unknown Station";
        }
    }
}
```

GasStation Class
The GasStation class models the data for a gas station. It contains properties such as the name and address, and a list of reviews.

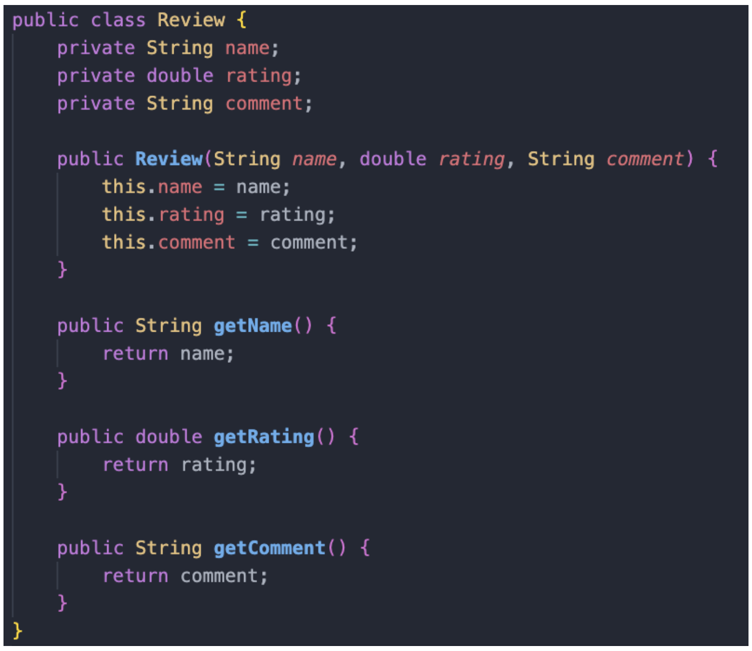
Constructors and Methods: Includes methods to retrieve the name, address, and reviews of a gas station, and a method to add a review.



```java
import java.util.ArrayList;
import java.util.List;

public class GasStation {
    private String name;
    private String address;
    private List<Review> reviews;

    public GasStation(String name, String address) {
        this.name = name;
        this.address = address;
        this.reviews = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public List<Review> getReviews() {
        return reviews;
    }

    public void addReview(Review review) {
        reviews.add(review);
    }
}
```

Review Class
The Review class represents a user's review of a gas station. It includes the reviewer's name, rating, and their written comment.

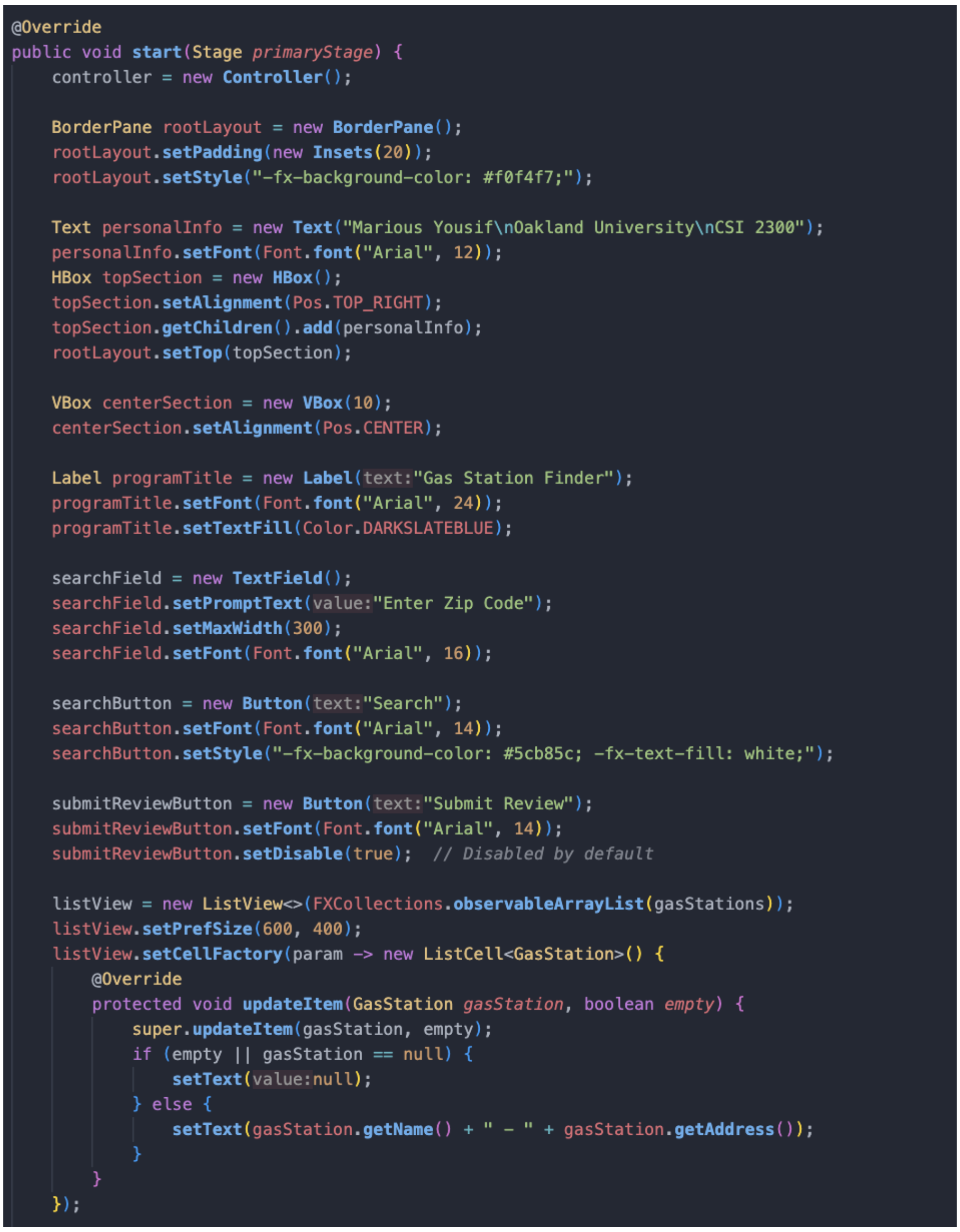
Constructors and Methods: Includes methods to retrieve the name, rating, and comment of a review.



```
public class Review {
    private String name;
    private double rating;
    private String comment;

    public Review(String name, double rating, String comment) {
        this.name = name;
        this.rating = rating;
        this.comment = comment;
    }

    public String getName() {
        return name;
    }

    public double getRating() {
        return rating;
    }

    public String getComment() {
        return comment;
    }
}
```

Main Class
The Main class is the entry point of the application. It sets up the JavaFX user interface and contains the event handlers for the interactive elements.

User Interface Setup: Defines the layout and interactive elements such as the text field, buttons, and list view.
Event Handlers: Includes handlers for search button action, submitting reviews, and displaying reviews.
Insert image of Main.java lines 2-211 here.



```java
@Override
public void start(Stage primaryStage) {
    controller = new Controller();

    BorderPane rootLayout = new BorderPane();
    rootLayout.setPadding(new Insets(20));
    rootLayout.setStyle("-fx-background-color: #f0f4f7;");

    Text personalInfo = new Text("Marious Yousif\nOakland University\nCSI 2300");
    personalInfo.setFont(Font.font("Arial", 12));
    HBox topSection = new HBox();
    topSection.setAlignment(Pos.TOP_RIGHT);
    topSection.getChildren().add(personalInfo);
    rootLayout.setTop(topSection);

    VBox centerSection = new VBox(10);
    centerSection.setAlignment(Pos.CENTER);

    Label programTitle = new Label(text:"Gas Station Finder");
    programTitle.setFont(Font.font("Arial", 24));
    programTitle.setTextFill(Color.DARKSLATEBLUE);

    searchField = new TextField();
    searchField.setPromptText(value:"Enter Zip Code");
    searchField.setMaxWidth(300);
    searchField.setFont(Font.font("Arial", 16));

    searchButton = new Button(text:"Search");
    searchButton.setFont(Font.font("Arial", 14));
    searchButton.setStyle("-fx-background-color: #5cb85c; -fx-text-fill: white;");

    submitReviewButton = new Button(text:"Submit Review");
    submitReviewButton.setFont(Font.font("Arial", 14));
    submitReviewButton.setDisable(true);  // Disabled by default

    listView = new ListView<>(FXCollections.observableArrayList(gasStations));
    listView.setPrefSize(600, 400);
    listView.setCellFactory(param -> new ListCell<GasStation>() {
        @Override
        protected void updateItem(GasStation gasStation, boolean empty) {
            super.updateItem(gasStation, empty);
            if (empty || gasStation == null) {
                setText(value:null);
            } else {
                setText(gasStation.getName() + " - " + gasStation.getAddress());
            }
        }
    });
```

Building and Running

The application can be compiled and run using any Java IDE that supports JavaFX. Ensure the JavaFX libraries are included in your project's build path.