SWE 645 – Assignment 3 README

# Group Name: Straw Hats

Group Members:

* Jhansi Sneha Kamsali (G01478467)
* Sheetal Sachidananda Harshini (G01464307)
* Gauthami Shravya Veerababu (G01461233)

# Objective

To develop a Microservices-based Spring Boot application for student surveys, perform CRUD operations using MySQL (Amazon RDS), containerize the application using Docker, deploy it on Kubernetes via Rancher, and automate the CI/CD process using Jenkins and GitHub.

# PART 1: Spring Boot Microservice

Executed by: Jhansi Sneha Kamsali

## Project Setup using Spring Initializr

1. Visit <https://start.spring.io>
2. Create a new Maven project with the required dependencies. Download the zip file and extract it.
3. Configure the following project settings:  
    - Project: Maven  
    - Language: Java  
    - Spring Boot: 3.x  
    - Group: edu.gmu.swe645  
    - Artifact: surveyservice  
    - Packaging: Jar  
    - Java version: 17  
    - Dependencies: Spring Web, Spring Data JPA, MySQL Driver, Lombok (optional)  
   3. Click "Generate", extract the .zip and open in IntelliJ or Eclipse.

## Project Structure and Endpoints

- **StudentSurvey.java**: Defines the JPA entity representing the structure of the student survey table in the database.

A screenshot of a computer

AI-generated content may be incorrect.

- **StudentSurveyController.java**: Handles REST API endpoints for survey operations like submit, retrieve, update, and delete.

A screenshot of a computer

AI-generated content may be incorrect.

- **StudentSurveyRepository.java**: Provides database access methods by extending Spring Data JPA's JpaRepository.

A screenshot of a computer

AI-generated content may be incorrect.

- **StudentSurveyServiceApplication.java**: The main Spring Boot application class that starts the service.

A screenshot of a computer

AI-generated content may be incorrect.

CRUD Operations:  
- Create a new survey : POST /api/surveys  
- View all surveys: GET /api/surveys  
- View individual survey by surveyId : GET /api/surveys/{id}  
- Update or edit an existing survey : PUT /api/surveys/{id}  
- Delete a survey : DELETE /api/surveys/{id}

The following are the endpoints to test the api’s in local using Postman (Installed Postman to test the endpoints in local)

Create (POST) - http://localhost:8080/api/surveys

View all (GET) - http://localhost:8080/api/surveys

View by id (GET) - http://localhost:8080/api/surveys/{id}

Update by using id (PUT) - http://localhost:8080/api/surveys/{id}

Delete a survey (DELETE) - http://localhost:8080/api/surveys/{id}

# PART 2: MySQL Database on Amazon RDS

Executed by: Jhansi Sneha Kamsali

1. **Accessed AWS RDS Console**

* Navigated to AWS Console and searched for the **RDS** service.
* Selected **"Create database"** from the dashboard.

1. **Created the Database**

* Used the **Standard Create** option.
* Selected **MySQL** as the engine.
* Chose the **Free Tier** template.
* Set **DB instance identifier** as surveydb.
* Set **master username** as admin and password as admin2025.

1. **Configured Instance Settings**

* Selected instance class: **db.t3.micro** (eligible under Free Tier).
* Used default storage configuration (20 GB).

1. **Set Connectivity Settings**

* Selected the **default VPC** and **default subnet group**.
* Set **public access** to **Yes**.
* Chose an **existing security group**: 645-security-group (sg-0cdbf1e80e4d795b1).

1. **Configured Database Settings**

* Enabled **password authentication**.
* Set **initial database name** as surveydb.

1. **Created the Database**

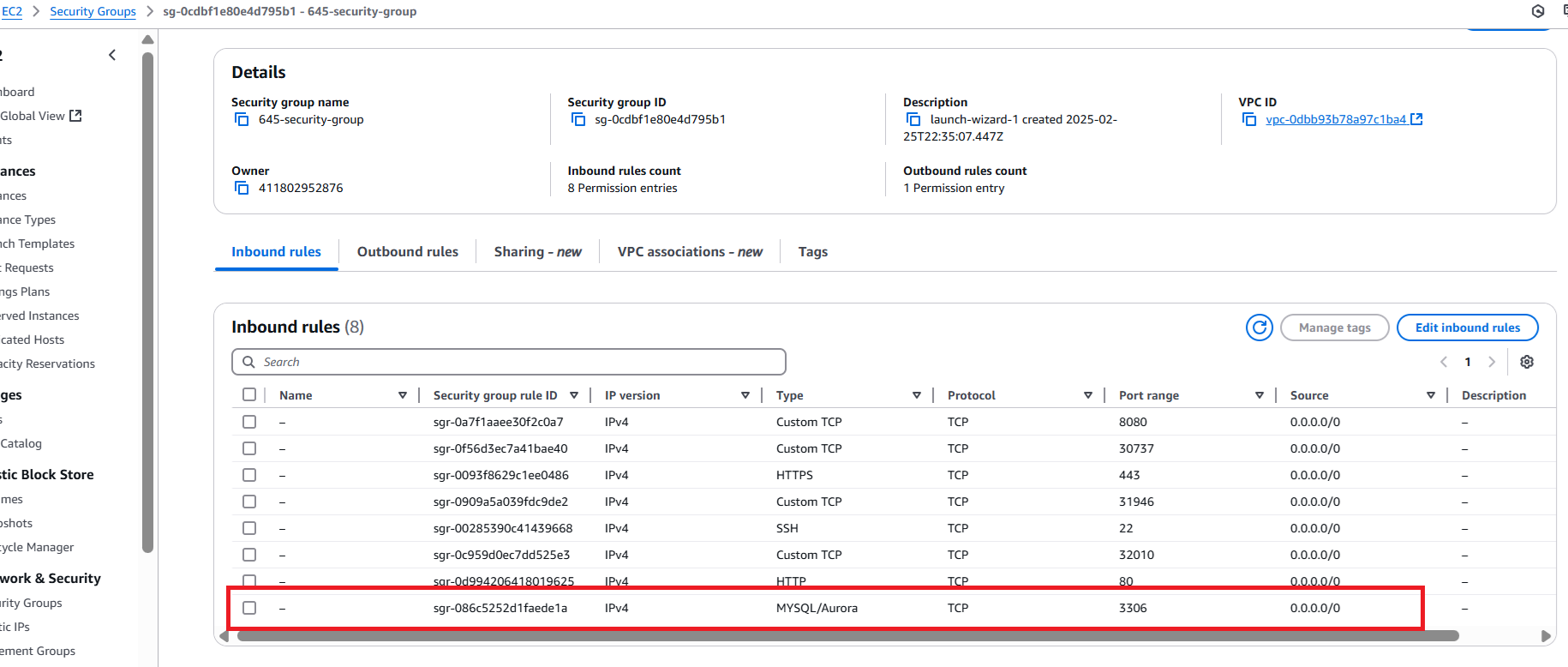
* Clicked **“Create Database”** and waited until the status changed to **Available**.

1. **Reviewed Endpoint & Networking**

* Navigated to the **Connectivity & security** tab.
* Endpoint noted as: surveydb.cslzkqywwmbl.us-east-1.rds.amazonaws.com
* Port: 3306

1. Security Group Configuration

Security group 645-security-group was **initially not configured** to allow inbound access on port 3306. Edited the inbound rules manually to **add a new rule** for **MySQL/Aurora (port 3306)** with **source set to 0.0.0.0/0** (accessible from anywhere).



1. **Used RDS in Application**

* In the Spring Boot application's application.properties (or environment variables in the deployment YAML), the following values were added:

A screenshot of a computer

AI-generated content may be incorrect.

**MySQL Workbench Download and RDS Connection Setup**

1. Open the MySQL Workbench download page:  
   👉 <https://dev.mysql.com/downloads/workbench/>
2. Download the version that matches the operating system being used (e.g., Windows 64-bit).
3. Install the downloaded package and launch **MySQL Workbench**.
4. Click on **"MySQL Connections" > "+"** to set up a new connection.
5. In the **"Setup New Connection"** window, enter the following:
   * **Connection Name**: surveydb-rds
   * **Hostname**: surveydb.cslzkqywwmbl.us-east-1.rds.amazonaws.com
   * **Port**: 3306
   * **Username**: admin (or whatever was configured during RDS setup)
   * **Password**: Click **Store in Vault** and enter admin2025 (or the actual password set earlier)
6. Test the connection using the **"Test Connection"** button. If successful, click **OK** to save the connection.
7. Open the saved connection from the Workbench home screen to access and manage the surveydb instance.
8. After connecting successfully to the RDS instance using MySQL Workbench:
9. Open the saved connection to surveydb-rds.
10. In the SQL Editor, create the required database by running the following SQL command: CREATE DATABASE surveydb;
11. Confirm the database was created by clicking the refresh icon in the Schemas section (left sidebar). surveydb should now appear in the list.

A screenshot of a computer login box

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# PART 3: Docker Containerization

Executed by: Gauthami Shravya Veerababu

Steps:  
1. Build: ./mvnw clean package

2. **Signed into DockerHub**

* Logged into <https://hub.docker.com> using DockerHub credentials.
* Installed **Docker Desktop** and signed in using the same DockerHub account.
  1. **Created a Dockerfile**
* In the root directory of the Spring Boot project, created a file named Dockerfile (without any extension) with the following content:

A screenshot of a computer

AI-generated content may be incorrect.

* 1. **Logged into Docker from the terminal**
* Used the following command to authenticate on the command line: docker login -u jhansisneha
* Entered the password when prompted to successfully log in.
  1. **Built the Docker image**
* Ran the following command to build the image using the Dockerfile:
  + docker build -t jhansisneha/swe645-hw3:latest .

A screenshot of a computer screen

AI-generated content may be incorrect.

* 1. **Ran the Docker container locally**

1. Verified the image by running it as a container:
   * docker run -p 8080:8080 jhansisneha/swe645-hw3:latest
2. Confirmed the Spring Boot app started successfully and responded at http://localhost:8080/api/surveys.
   1. **Pushed the image to DockerHub**

* Pushed the image using the following command
  + docker push jhansisneha/swe645-hw3:latest
  1. **Verified the push on DockerHub**
* Logged into [DockerHub](https://hub.docker.com) and navigated to the **Repositories** tab.
* Confirmed that the image swe645-hw3 appeared under the account jhansisneha with the latest tag.

A screenshot of a computer

AI-generated content may be incorrect.  
**PART 4: Kubernetes Deployment via Rancher**

Executed by: Gauthami Shravya Veerababu

1. Logged into the AWS Learner Lab and accessed the EC2 Instances page.

2. Previously created three EC2 instances for Rancher, Kubernetes, and Jenkins during Assignment 2, with the following settings:

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Inbound traffic allowed on ports 8080, 80, 443, and 22

Each instance had 30 GB storage

Outbound rules configured to allow all traffic

Elastic IPs assigned to all three instances for persistent access

Docker already installed on both Rancher and Kubernetes instances from the earlier assignment.

Rancher UI was already up and running on Instance 1 from the previous assignment. No need to restart the Rancher container.

* + 1. Instead of creating a new cluster, used the existing Rancher-managed cluster (hw2-cluster) to deploy the new container for Assignment 3.

Deployment Steps

* Logged into the Rancher UI using the previously set custom admin password.
* Opened the existing cluster (hw2-cluster) created during Assignment 2.
* Navigated to Workloads → Deployments, and created a new deployment for Assignment 3:
  + Deployment Name: swe645-hw3
  + Container Image: jhansisneha/swe645-hw3:latest
  + **Replicas**: 3 (to ensure high availability with three pods running)
  + Private container port: 8080
  + Number of replicas: 1
  + Chose NodePort service type
* Clicked Deploy to create the workload and deploy the application.
* Confirmed that **three pods were running** successfully under the deployment swe645-hw3.

A screenshot of a computer

AI-generated content may be incorrect.

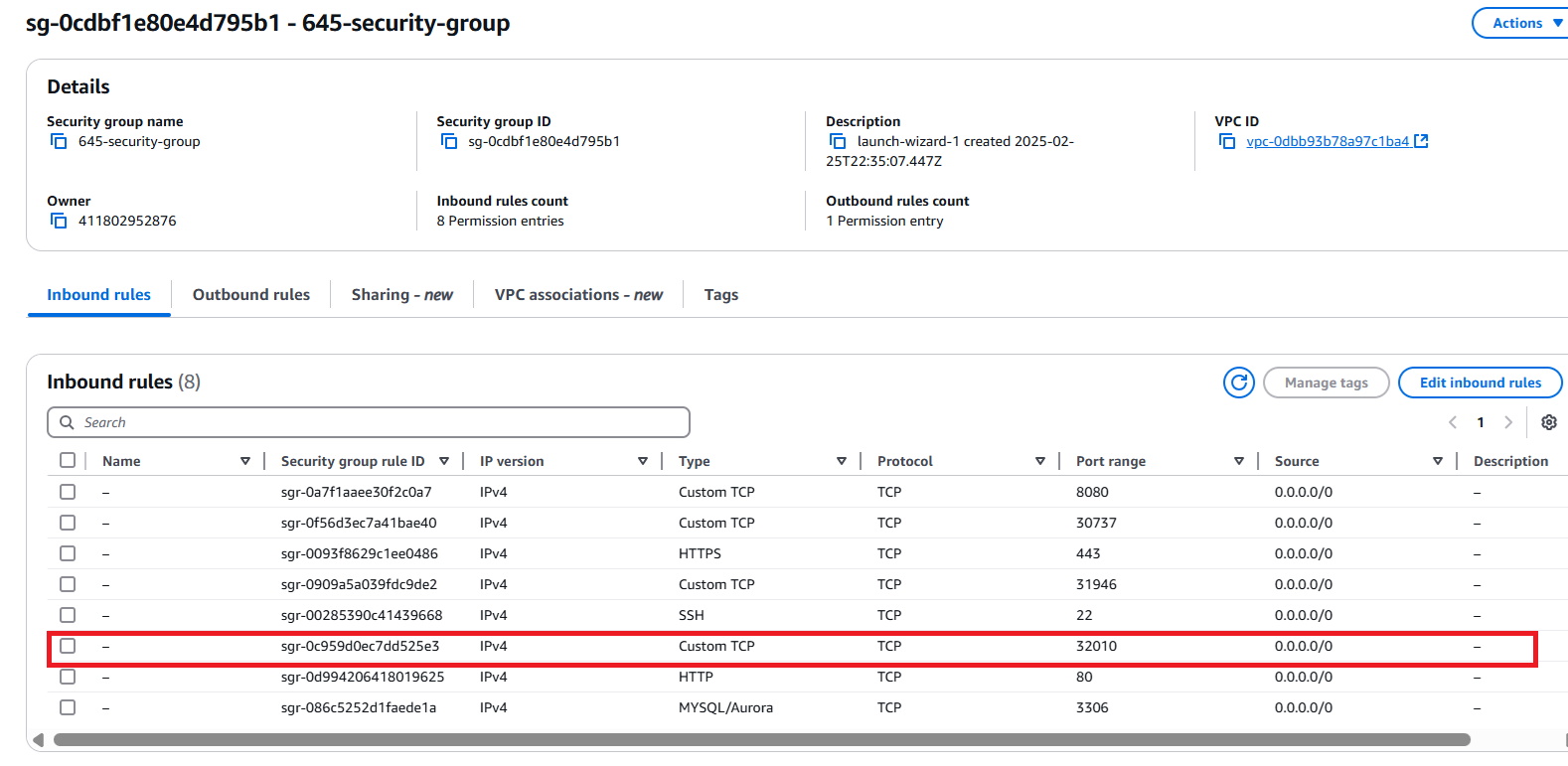
To verify access:

* + 1. Ran the following command in the Rancher cluster shell to get the assigned NodePort:

kubectl get service

Noted the NodePort mapped to survey-service.

* + 1. Updated the EC2 security group inbound rules (for Kubernetes instance) to allow traffic on the newly assigned NodePort (e.g., 32010) from anywhere (0.0.0.0/0)`.



* + 1. Verified deployment by accessing the app using:

http://<Public DNS of Kubernetes Instance>:<NodePort>/api/surveys

i.e., http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys

A screenshot of a computer

AI-generated content may be incorrect.

# PART 5: CI/CD with Jenkins

Executed by: Sheetal Sachidananda Harshini

1. Jenkins already installed on EC2 Instance 3 during previous assignments, along with required components:
   * Docker
   * Java Development Kit (JDK)
   * Maven
2. **Jenkins UI Access**
   * Accessed Jenkins through: http://<Public IPv4 of Instance 3>:8080
3. **Plugin** Installation
   * From Jenkins dashboard: Manage Jenkins → Manage Plugins  
     Installed the following plugins:
     + Git plugin
     + Docker Pipeline
     + Kubernetes CLI
     + Pipeline: Stage View
4. **Credentials Configuration**
   * From Jenkins dashboard:  
     → Manage Jenkins → Credentials → Global → Add Credentials
   * Added:
     + **GitHub credentials**
       - Type: Username with password
       - ID: git
     + **DockerHub credentials**
       - Type: Username with password
       - ID: docker
     + **Kubeconfig file**
       - Type: Secret file
       - ID: kube\_config

A screenshot of a computer

AI-generated content may be incorrect.

1. **Configured Jenkins to Use Docker**
   * Added Jenkins to the Docker group to allow Docker CLI commands:

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

1. **Created Jenkins Pipeline for CI/CD**
   * From Jenkins dashboard: → New Item → Pipeline
   * Named pipeline: hw3-cicdpipeline
   * Configured:
     + **Build Trigger**: Enabled *Poll SCM* with schedule \* \* \* \* \*
     + **Pipeline Script from SCM**:
       - SCM: Git
       - Repository: <https://github.com/jhansisneha/645-hw3>
       - Branch: main
       - Script path: Jenkinsfile
2. **Jenkinsfile Structure**
   * The Jenkinsfile used for automation is stored in the GitHub repository.

**Jenkinsfile:**

pipeline {

agent any

environment {

IMAGE = 'jhansisneha/swe645-hw3:latest'

}

stages {

stage('Checkout') {

steps {

checkout scm

}

}

stage('Build') {

steps {

sh './mvnw clean package -DskipTests'

}

}

stage('Docker Build & Push') {

steps {

withCredentials([usernamePassword(credentialsId: 'docker', usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {

sh """

docker build -t $IMAGE .

echo "$DOCKER\_PASS" | docker login -u "$DOCKER\_USER" --password-stdin

docker push $IMAGE

"""

}

}

}

stage('Deploy to Kubernetes') {

steps {

withCredentials([file(credentialsId: 'kube\_config', variable: 'KUBECONFIG\_FILE')]) {

sh """

mkdir -p ~/.kube

cp \$KUBECONFIG\_FILE ~/.kube/config

chmod 600 ~/.kube/config

kubectl apply -f k8s/survey-deployment.yaml

kubectl apply -f k8s/survey-service.yaml

"""

}

}

}

}

}

* + It includes the following stages:

**Testing the CI/CD Pipeline**

* + Triggered the pipeline by committing code to the GitHub repository.
  + Jenkins automatically:
    - Pulled the latest code
    - Built the Spring Boot project
    - Created and pushed the Docker image
    - Applied the deployment and service to the existing Rancher-managed Kubernetes cluster

A screenshot of a computer

AI-generated content may be incorrect.

# Final Outputs and Links

- GitHub Repo: https://github.com/jhansisneha/645-hw3  
- DockerHub: <https://hub.docker.com/r/jhansisneha/swe645-hw3>

- Rancher: <https://3.232.98.81/dashboard/auth/login>

- Jenkins: <http://54.224.88.206:8080/login?from=%2F>  
- **APIs:**

**URLS:**

Create Survey (POST) - <http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys>

View all Surveys (GET) - <http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys>

View Survey by id (GET) - [http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/{id}](http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/%7bid%7d)

Update Survey by using id (PUT) - [http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/{id}](http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/%7bid%7d)

Delete a survey (DELETE) - [http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/{id}](http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/%7bid%7d)

**Sample Requests and Responses:**

**1. Getting all surveys**

GET: <http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys>

Sample Request:

GET /api/surveys HTTP/1.1

Host: ec2-3-88-76-65.compute-1.amazonaws.com:32010

Sample Response:

[{"id":1,"firstName":"Sheetal","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"},{"id":3,"firstName":"Jane 6","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"},{"id":7,"firstName":"Jane 6","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"},{"id":9,"firstName":"Jane 8","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"},{"id":10,"firstName":"Jane 8","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"}]

**2.Getting a survey by id**

Sample Request:

GET [http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/1](http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/1200)

GET /api/surveys/1 HTTP/1.1

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: 9e3ad21f-5926-44df-ad01-b2f6a07b3803

Host: ec2-3-88-76-65.compute-1.amazonaws.com:32010

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Sample Response:

HTTP/1.1 200 OK

Content-Type: application/json

Transfer-Encoding: chunked

Date: Sun, 13 Apr 2025 18:20:36 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{"id":1,"firstName":"Sheetal","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"}

**3.Creating a survey**

Sample Request:

POST /api/surveys HTTP/1.1

Host: ec2-3-88-76-65.compute-1.amazonaws.com:32010

Content-Type: application/json

Content-Length: 356

{

"firstName": "Jane 8",

"lastName": "Doe",

"streetAddress": "123 Campus Drive",

"city": "Fairfax",

"state": "VA",

"zip": "22030",

"telephone": "123-456-7890",

"email": "jane.doe@example.com",

"dateOfSurvey": "2025-03-26",

"likedMost": "campus",

"interestSource": "internet",

"recommendationLikelihood": "Very Likely"

}

Sample Response:

HTTP/1.1 200 OK

Content-Type: application/json

Transfer-Encoding: chunked

Date: Sun, 13 Apr 2025 18:11:58 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{"id":11,"firstName":"Jane 8","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"}

**4.Updating a Survey**

Sample Request:

PUT <http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/1200>

PUT /api/surveys/1 HTTP/1.1

Content-Type: application/json

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: aab81b91-cddf-4139-b237-e46d51f986c6

Host: ec2-3-88-76-65.compute-1.amazonaws.com:32010

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Content-Length: 359

{

"firstName": "Sheetal12",

"lastName": "Doe",

"streetAddress": "123 Campus Drive",

"city": "Fairfax",

"state": "VA",

"zip": "22030",

"telephone": "123-456-7890",

"email": "jane.doe@example.com",

"dateOfSurvey": "2025-03-26",

"likedMost": "campus",

"interestSource": "internet",

"recommendationLikelihood": "Very Likely"

}

Sample Response:

HTTP/1.1 200 OK

Content-Type: application/json

Transfer-Encoding: chunked

Date: Sun, 13 Apr 2025 18:22:03 GMT

Keep-Alive: timeout=60

Connection: keep-aliv

{"id":1,"firstName":"Sheetal12","lastName":"Doe","streetAddress":"123 Campus Drive","city":"Fairfax","state":"VA","zip":"22030","telephone":"123-456-7890","email":"jane.doe@example.com","dateOfSurvey":"2025-03-26","likedMost":"campus","interestSource":"internet","recommendationLikelihood":"Very Likely"}

**5.Deleting a Survey**

Sample Request:

DELETE <http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/1200>

DELETE /api/surveys/1 HTTP/1.1

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: 3b1d613f-4128-4dae-9c04-8d8135e97c82

Host: ec2-3-88-76-65.compute-1.amazonaws.com:32010

Accept-Encoding: gzip, deflate, br

Connection: keep-alive