# SWE 645 Extra Credit Readme File

**Group Name:** Straw Hats

**Group Members**:

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

## Part 1: Creating the Python FastAPI Microservice

**Executed by:** Jhansi Sneha Kamsali

**1. Set up the Development Environment**

* Created a project directory on my local machine.
* Created a Python virtual environment to isolate dependencies.

*conda create --name swe645 python=3.11*

*conda activate swe645*

* Installed required Python packages for backend development.

*pip install fastapi uvicorn sqlalchemy pymysql*

* + fastapi: Framework for building APIs quickly.
  + uvicorn: ASGI server to run the FastAPI app.
  + sqlalchemy: ORM to interact with MySQL database.
  + pymysql: MySQL client to enable SQLAlchemy to talk to MySQL.

**2. Designed the Database Model**

* Created a file called models.py.
* Defined a StudentSurvey model using SQLAlchemy ORM.
* The model maps to a MySQL table with the following fields:
  + id: Primary Key (auto-increment)
  + first\_name, last\_name
  + street\_address, city, state, zip
  + telephone, email
  + date\_of\_survey
  + liked\_most
  + interest\_source
  + recommendation\_likelihood
* **models.py:**

A screenshot of a computer

AI-generated content may be incorrect.

**3. Configured Database Connection**

* Created a file called database.py.
* Configured SQLAlchemy to connect to Amazon RDS MySQL using the pymysql driver.
* **Database URL format:** mysql+pymysql://<username>:<password>@<db-endpoint>:3306/<database-name>
* **database.py**

A screenshot of a computer

AI-generated content may be incorrect.

**4. Implemented CRUD Operations**

* Created a file called crud.py.

A screenshot of a computer screen

AI-generated content may be incorrect.

* Wrote modular CRUD functions:
  + create\_survey(db, survey)
  + get\_all\_surveys(db)
  + get\_survey(db, survey\_id)
  + update\_survey(db, survey\_id, survey\_data)
  + delete\_survey(db, survey\_id)

**5. Built FastAPI Endpoints**

* Created main.py to define the FastAPI application.

A screenshot of a computer program

AI-generated content may be incorrect.

* Imported models, database, and crud modules.
* Set up endpoints:
  + **POST** /api/surveys → Create new survey
  + **GET** /api/surveys → Retrieve all surveys
  + **GET** /api/surveys/{survey\_id} → Retrieve survey by ID
  + **PUT** /api/surveys/{survey\_id} → Update survey by ID
  + **DELETE** /api/surveys/{survey\_id} → Delete survey by ID
* Used dependency injection to pass the DB session.

**7. Tested Locally**

* Ran the app using Uvicorn: uvicorn main:app --host 0.0.0.0 --port 8080
* Accessed FastAPI's built-in Swagger UI at http://localhost:8080/docs
* Tested all CRUD operations using Swagger and Postman.

**Directory Structure**

/ExtraCredit

|-- crud.py

|-- database.py

|-- main.py

|-- models.py

|-- requirements.txt

A screenshot of a computer

AI-generated content may be incorrect.

## Part 2: Setting up Amazon RDS MySQL for the Python Microservice

**Executed by:** Jhansi Sneha Kamsali

**1. Launch a New Amazon RDS Instance**

* Opened the AWS Management Console.
* Navigated to Aurora and RDS > Create database.
* Chose the following settings:
  + **Engine type**: MySQL
  + **Version**: MySQL 8.0
  + **Templates**: Free tier
  + **DB instance identifier**: surveydb
  + **Master username**: admin
  + **Master password**: admin2025
  + **Public access**: Enabled to allow connection from EC2 and local dev
  + **VPC security group**: Selected a group allowing port 3306 inbound

**2. Modify the Security Group**

* Edited the security group attached to the RDS instance:
  + **Inbound rule**:
    - Type: MySQL/Aurora
    - Port Range: 3306
    - Source: 0.0.0.0/0

**3. Create the Database**

* After the RDS instance was ready:
  + Copied the RDS endpoint from the AWS console
  + Connected to it using MySQL Workbench with:
    - Host: surveydb.cma9bsrjawrk.us-east-1.rds.amazonaws.com
    - Port: 3306
    - User: admin
    - Password: admin2025
* Created a new database: CREATE DATABASE surveydb;

**4. Update the Python App to Connect to RDS**

* Edited database.py in the Python project:

SQLALCHEMY\_DATABASE\_URL = "mysql+pymysql://admin:admin2025surveydb.cma9bsrjawrk.us-east-1.rds.amazonaws.com:3306/surveydb"A screenshot of a computer

AI-generated content may be incorrect.

* Replaced <rds-endpoint> with the actual host provided by AWS RDS.
* This made all API calls connect directly to the cloud-hosted MySQL database.

**5. Verify Connection**

* Ran the FastAPI app locally (uvicorn main:app) and sent a POST /api/surveys request.
* Checked that the data was saved into the student\_surveys table in surveydb (RDS).
* Verified using SQL queries in MySQL Workbench.

A screenshot of a computer

AI-generated content may be incorrect.

## Part 3: Dockerizing the Python FastAPI Application

**Executed by:** Jhansi Sneha Kamsali

**Step 1: Create a Dockerfile**

In the root directory of the FastAPI project, created a file named Dockerfile with the following content:

A screenshot of a computer

AI-generated content may be incorrect.

**Explanation:**

* **Base image**: Uses a minimal Python 3.11 image.
* **Working directory**: Sets the container's root to /app.
* **COPY**: Adds all project files into the container.
* **pip install**: Installs all packages from requirements.txt.
* **EXPOSE**: Opens port 8080 so Kubernetes can forward traffic.
* **CMD**: Launches the app using Uvicorn.

**Step 2: Create requirements.txt**

Listed all project dependencies:

fastapi

uvicorn

sqlalchemy

pymysql

A screenshot of a computer

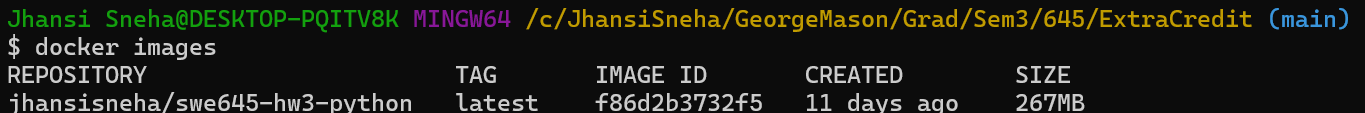
AI-generated content may be incorrect.

**Step 3: Build the Docker Image**

From the root of the project directory, ran:

*docker build -t jhansisneha/swe645-hw3-python:latest .*

This created a local image tagged swe645-hw3-python.



**Step 4: Test the Docker Image Locally (Optional)**

*docker run -p 8080:8080 jhansisneha/swe645-hw3-python:latest*

Confirmed the app was accessible at http://localhost:8080/docs.

**Step 5: Push to DockerHub**

Logged in and pushed the image

*docker login -u jhansisneha*

*docker push jhansisneha/swe645-hw3-python:latest*

The Docker image is now publicly available and ready for deployment on Kubernetes.

A screenshot of a computer

AI-generated content may be incorrect.

## Part 4: Deploying to Kubernetes using Rancher

**Executed by:** Sheetal Sachidananda Harshini

Deploy the Dockerized FastAPI application to a Kubernetes cluster using Rancher hosted on AWS EC2, ensuring scalability, manageability, and high availability.

**1. Launch EC2 Instances (AWS)**

| **Instance** | **Use** |
| --- | --- |
| **Instance 1** | Rancher Server |
| **Instance 2** | Kubernetes Node |
| **Instance 3** | Jenkins Server (for Part 5) |

**Common Configuration:**

* **AMI**: Ubuntu 22.04 LTS (HVM), SSD
* **Storage**: 30GB
* **Ports Opened**: 22 (SSH), 80, 443, 8080
* **Elastic IPs** assigned to each instance

**2. Install Docker and Rancher on Instance 1**

*sudo apt update*

*sudo apt install docker.io -y*

*sudo systemctl start docker*

*sudo systemctl enable docker*

**Run Rancher:** *sudo docker run -d --restart=unless-stopped \ -p 80:80 -p 443:443 --privileged rancher/rancher*

* Access Rancher at: https://<Instance1-IP> i.e., <https://13.217.73.187>
* Set custom admin password.
  + In the UI, copy the password command and paste in instance1 console to get the default password to enter in the RancherUI for the first time login. The noted container-ID must be replaced in the below command

*sudo docker logs container-ID 2>&1 | grep “Bootstrap Password:”*

* + Use the generated password to login and once login, select the option “Set a specific password to use” to set a custom password for future.
* Rancher UI auto-generates a **node registration command** in the next step.

**3. Register Kubernetes Node (Instance 2)**

* Once custom password is set navigate to home and create a new cluster by clicking the create button and selecting custom under “Use existing nodes and create and cluster using RKE”. Give the cluster a name and click Create. Selected all three roles: etcd, Control Plane, Worker and apply the checkbox – Insecure: Select this to skip TLS verification if your server has a self-signed certificate. This would give a registration command to be pasted on instance2 connect console.
* Copied the registration command and ran it on Instance 2.

*curl --insecure -fL https://13.217.73.187/system-agent-install.sh | sudo sh -s - \*

*--server https://13.217.73.187 --label 'cattle.io/os=linux' \*

*--token <auth-token> --etcd --controlplane --worker*

* Waited for the cluster to show Active status in Rancher UI.

**4. Deploy App on Kubernetes**

**Used the following YAMLs:**

**survey-deployment.yaml**

A screenshot of a computer

AI-generated content may be incorrect.

**survey-service.yaml**

A screenshot of a computer

AI-generated content may be incorrect.

* Create a deployment object using the Docker container image.
* After that, Click the cluster name -> from left side menu open ‘Workloads’ and select ‘Deployments’.
* Click on ‘Create’ button and fill the following fields
  + Fill the name field for the deployment object.
  + Increase count in replica field from 1 to 3.
  + Fill the container field.
  + Fill Container image field. (it would be the one created in docker)
  + Click Add Port or Service and give Nodeport in the Service type and 80 in the private container port field. 80 because in the docker file I have used value 80 for EXPOSE. Click Create button.
* **Access Application:**

You can access the application using the following format:

- `<Public DNS IP of instance 2>:<assigned NodePort>`

<http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/>

A screenshot of a computer

AI-generated content may be incorrect.

## Part 5: CI/CD Pipeline with Jenkins

**Executed by:** Gauthami Shravya Veerababu

**1. Set Up Jenkins on EC2**

* Launched a new Ubuntu 22.04 EC2 instance.
* Installed Jenkins and Docker:

*sudo apt update*

*sudo apt install openjdk-17-jdk -y*

*wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -*

*sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'*

*sudo apt update*

*sudo apt install jenkins -y*

*sudo systemctl start jenkins*

*sudo usermod -aG docker jenkins*

*sudo systemctl restart jenkins*

* Opened Jenkins at:

http://<jenkins-ec2-public-ip>:8080

<http://98.85.187.92:8080/>

* Unlocked Jenkins using the password from:

*sudo cat /var/lib/jenkins/secrets/initialAdminPassword*

**2. Install Required Plugins**

Installed the following plugins via Manage Jenkins > Plugins:

* Git plugin
* Docker Pipeline
* Kubernetes CLI Plugin
* Pipeline Utility Steps

**3. Add Required Credentials**

Via Manage Jenkins > Credentials, added:

| **ID** | **Type** | **Used For** |
| --- | --- | --- |
| git | Username+Password | Accessing private GitHub repo |
| docker | Username+Password | DockerHub login to push images |
| kube\_config | Secret File | kubeconfig to connect to K8s cluster |

**4. Create a Pipeline**

* Created a Pipeline job named cicdPipeline.
* Selected “Pipeline script from SCM”:
  + SCM: Git
  + Repo: <https://github.com/jhansisneha/645-extra-credit>
  + Script Path: Jenkinsfile

**5. Jenkinsfile Used**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**6. Triggering the Pipeline**

* Configured SCM polling (\* \* \* \* \*) to check GitHub every minute.

**7. Result**

* Upon pushing to GitHub:
  + Jenkins cloned the repo
  + Built the Docker image
  + Pushed it to DockerHub
  + Updated the image in Kubernetes using kubectl set image

A screenshot of a computer

AI-generated content may be incorrect.

**Final Outputs and Links**

- GitHub Repo: <https://github.com/jhansisneha/645-extra-credit.git>  
- Docker Hub: <https://hub.docker.com/repository/docker/jhansisneha/swe645-hw3-python/general>

- Rancher: <https://13.217.73.187/dashboard>

- Jenkins: [http://98.85.187.92:8080//login?from=%2F](http://54.224.88.206:8080/login?from=%2F)  
- **APIs:**

**URLS:**

Create Survey (POST) - <http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys>

View all Surveys (GET) - [http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys](http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys)

View Survey by id (GET) - [http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/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://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys/{id}](http://ec2-3-88-76-65.compute-1.amazonaws.com:32010/api/surveys/%7bid%7d)

Delete a survey (DELETE) - [http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/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://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys>

Sample Request:

GET /api/surveys HTTP/1.1

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: d25b9ff4-d2ad-4733-904e-3eb64a88bd29

Host: ec2-54-243-106-23.compute-1.amazonaws.com:30514

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Sample Response:

HTTP/1.1 200 OK

date: Tue, 06 May 2025 19:26:49 GMT

server: uvicorn

content-length: 630

content-type: application/json

[{"city":"UpdatedCity","street\_address":"456 Updated St","zip":"22201","email":"updated@gmu.edu","liked\_most":"Atmosphere","recommendation\_likelihood":"Likely","last\_name":"Kamsali","id":1,"first\_name":"UpdatedFirstName","state":"VA","telephone":"9876543210","date\_of\_survey":"2025-04-26","interest\_source":"Internet"},{"city":"UpdatedCity","street\_address":"123 College St","zip":"22030","email":"js@gmu.edu","liked\_most":"Campus","recommendation\_likelihood":"Very Likely","last\_name":"Sneha","id":3,"first\_name":"UpdatedFirstName","state":"VA","telephone":"1234567890","date\_of\_survey":"2025-04-25","interest\_source":"Friends"}]

**2.Getting a survey by id**

Sample Request:

GET <http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys/3>

GET /api/surveys/3 HTTP/1.1

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: 4e519acb-5f25-43df-806d-493acee81ed0

Host: ec2-54-243-106-23.compute-1.amazonaws.com:30514

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Sample Response:

HTTP/1.1 200 OK

date: Tue, 06 May 2025 19:31:26 GMT

server: uvicorn

content-length: 310

content-type: application/json

{"city":"UpdatedCity","street\_address":"123 College St","zip":"22030","email":"js@gmu.edu","liked\_most":"Campus","recommendation\_likelihood":"Very Likely","id":3,"last\_name":"Sneha","first\_name":"UpdatedFirstName","state":"VA","telephone":"1234567890","date\_of\_survey":"2025-04-25","interest\_source":"Friends"}

**3.Creating a survey**

Sample Request:

POST <http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys>

POST /api/surveys HTTP/1.1

Content-Type: application/json

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: e322078a-0ff0-4da6-a58e-a79dbf994f6d

Host: ec2-54-243-106-23.compute-1.amazonaws.com:30514

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Content-Length: 351

{

"first\_name": "Jhansi",

"last\_name": "Sneha",

"street\_address": "123 College St",

"city": "Fairfax",

"state": "VA",

"zip": "22030",

"telephone": "1234567890",

"email": "js@gmu.edu",

"date\_of\_survey": "2025-04-25",

"liked\_most": "Campus",

"interest\_source": "Friends",

"recommendation\_likelihood": "Very Likely"

}

Sample Response:

HTTP/1.1 200 OK

date: Tue, 06 May 2025 17:50:35 GMT

server: uvicorn

content-length: 296

content-type: application/json

{"city":"Fairfax","street\_address":"123 College St","zip":"22030","email":"js@gmu.edu","liked\_most":"Campus","recommendation\_likelihood":"Very Likely","id":3,"last\_name":"Sneha","first\_name":"Jhansi","state":"VA","telephone":"1234567890","date\_of\_survey":"2025-04-25","interest\_source":"Friends"}

**4.Updating a Survey**

Sample Request:

PUT <http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys/3>

PUT /api/surveys/3 HTTP/1.1

Content-Type: application/json

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: bd61e577-f92d-4b75-80f1-77fe93d636b4

Host: ec2-54-243-106-23.compute-1.amazonaws.com:30514

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Content-Length: 68

{

"first\_name": "UpdatedFirstName",

"city": "UpdatedCity"

}

Sample Response:

HTTP/1.1 200 OK

date: Tue, 06 May 2025 17:53:53 GMT

server: uvicorn

content-length: 310

content-type: application/json

{"city":"UpdatedCity","street\_address":"123 College St","zip":"22030","email":"js@gmu.edu","liked\_most":"Campus","recommendation\_likelihood":"Very Likely","id":3,"last\_name":"Sneha","first\_name":"UpdatedFirstName","state":"VA","telephone":"1234567890","date\_of\_survey":"2025-04-25","interest\_source":"Friends"}

**5.Deleting a Survey**

Sample Request:

DELETE <http://ec2-54-243-106-23.compute-1.amazonaws.com:30514/api/surveys/3>

DELETE /api/surveys/3 HTTP/1.1

User-Agent: PostmanRuntime/7.43.3

Accept: \*/\*

Postman-Token: 9f7c50c0-5b9d-4b1e-a8c6-dee7d70c85c9

Host: ec2-54-243-106-23.compute-1.amazonaws.com:30514

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Sample Response:

HTTP/1.1 200 OK

date: Tue, 06 May 2025 19:33:03 GMT

server: uvicorn

content-length: 21

content-type: application/json

{"message":"Deleted"}