

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Projects in Machine Learning

INFO8665 - Spring 2025 - Section 1

Prof. Anasuya Bhima

Group 4 - Section 1

Assignment 3

Due on June 11, 2025

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Assignment 3**

**1.1 DevOps Project Board**

**1.1.1 Project Board URL and Snapshots**

URL: <https://dev.azure.com/Vsharma3188/SageCare%202.0/_sprints/taskboard/SageCare%202.0%20Team/SageCare%202.0/Sprint%200>

A screenshot of a computer

AI-generated content may be incorrect.

*fig 1.1 Snapshot of team members including professor*

**1.2 Product Backlog Items – Groomed for sprint 1**

The below screenshot displays the groomed product backlog items for the sprint 1 from the Azure Dev board. Product backlog grooming took place several times during the sprint via sprint planning meetings and scrum meetings.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.3 Snapshot of the groomed product backlog with epics. Features, PBIs and tasks for the sprint 1*

The below screenshot shows the progress of each task of the sprint 1 from the Azure dev board.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.4 Snapshot of project board with tasks for sprint 1*

During the sprint planning, we created an “Internal Review” task for each Feature as shown in the below screenshot.

**A screenshot of a computer

AI-generated content may be incorrect.** *Fig 1.4 Snapshot of project board with internal review task for each feature in sprint 1*

We also created user acceptance testing for each task in the Azure dev board as shown in the screenshot below.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.5 Snapshot of task with User Acceptance criteria defined.*

**1.4 GitHub Repo and DevOps Board Integration**

**1.4.1 GitHub URL and Folder Structure**

URL: <https://github.com/CelebrityITPro/SageCare-2.0>

In the Github, it currently shows 3 branches i.e., main, sprint-0 and sprint-1 branches.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.6 Snapshot of GitHub folder structure*

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.7 Snapshot of GitHub Repo access*

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.8 GitHub Repo integration with Azure DevOps board*

**1.4.2 GitHub Pull request issue and approval**

The pull request was created to create sprint-1 branch and approval was required for this pull request. Once the approval is done, the pull request was merged successfully. The below screenshot shows the Pull request issue and approval from GitHub.

*A screenshot of a computer

AI-generated content may be incorrect.*

*Fig 1.9 GitHub pull request issue and approval*

**1.4.3 GitHub Sprint 1 branch with files uploaded.**

After the sprint-1 branch is created in the GitHub, the files were uploaded into the sprint-1 folder as shown in the below screenshots.

*A screenshot of a computer

AI-generated content may be incorrect.*

*Fig 1.10 GitHub Sprint 1 branch*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Fig 1.11 GitHub sprint 1 file updates*

**1.5 Team Roles and Responsibilities**

|  |  |  |
| --- | --- | --- |
| **Names** | **Roles** | **Responsibilities** |
| Ifediorah Kenechukwu | Project Manager | - Break down tasks and ensures alignment with vision - Work on project submission documents - Works on ML model development |
| Agu Jennifer | Scrum Master | - Ensures team follows agile process and all tasks are updated accordingly - Works on ML model development - Manages Azure DevOps board |
| Palakodeti Ravi Sumanth | Developer | - Works on ML model development - Explore and Preprocesses collected data and selects the best algorithms.  - Deploys models into production and ensures they perform reliably. |
| Sharma Vaibhav | Developer | - Designs the product frontend interface  - Develops backend and frontend architectures - Works with APIs, databases, and UI. |
| Owulu Amarachukwu | Software Tester | - Designs and runs test cases. - Explore and preprocesses collected data.  - Develops presentation slides. |

*Table 1.12 Team roles and responsibilities*

**1.6 Architecture**

**1.6.1 MLOps API designs**

The below section discusses about the API designs for each use case using RESTful HTTP method.

**1.6.1.1 Food Nutritional analysis**

Below screenshot show the RESTful API calls for **Food Nutritional Analysis** use case.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.13 RESTful API calls for Food Nutritional Analysis use case*

The screenshot below shows the “signup” service along with the parameters, protocol (POST) and response codes.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.14 Signup Service RESTful API call*

**1.6.1.2 NLP and Deep Learning for Free-Text Diagnosis**

Below screenshot show the RESTful API calls for **NLP and Deep Learning for Free-Text Diagnosis** use case.

A close-up of a list

AI-generated content may be incorrect.

*Fig 1.15 RESTful API calls for* ***NLP and Deep Learning for Free-Text Diagnosis*** *use case*

The screenshot below shows the “Retrieve Raw Symptom Entry” service along with the parameters, protocol (POST) and response codes.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.16 Register Service RESTful API call*

**1.6.1.3 Speech-to-Text for Doctor-Patient Consultation**

Below screenshot show the RESTful API calls for **Speech-to-Text for Doctor-Patient Consultation** use case.

**A screenshot of a computer

AI-generated content may be incorrect.**

*Fig 1.17 RESTful API calls for* ***Speech-to-Text for Doctor-Patient Consultation*** *use case*

The screenshot below shows the “Login” service along with the parameters, protocol (POST) and response codes.

A screenshot of a computer

AI-generated content may be incorrect.

*Fig 1.18 Login Service RESTful API call*

**1.6.2 Application Architecture**

A diagram of a software company

AI-generated content may be incorrect.

*Fig 1.19 Architecture diagram*

**1.6.2.1 Architecture Flow**

1. **Client & API Gateway**

* Every client request—from either the patient/doctor mobile or web app—arrives at a single API Gateway (api.sagecareapp.com:443) over HTTPS.
* The gateway then routes REST calls to three logical core service clusters

1. **Core Services:**
   * It includes Authentication, NLP and STT service.
   * It hosts both authentication and user-management endpoints alongside appointment, referral, and history operations, all sharing a single MongoDB application database (27017) to simplify data management.
2. **The “NLP Services”:**
   * Natural-language free text processing and symptom-interpretation endpoints are under one roof.
   * It communicates with the shared MongoDB store for storing parsed symptom text and any derived patient records.
3. **The “Speech to Text Services”** 
   * This implements speech-to-text features, feeding transcribed audio into the application database for downstream processing.
4. **Nutrition Service**:
   * It runs on its own endpoint but registers its inference model with the central Model Registry.
5. **Model Layer & Registry:**
   * All three model containers—diagnosis, transcription, and food—push their metadata to this registry over a uniform REST API (HTTP:5000), enabling the **Training** **Pipeline** to fetch current versions (HTTP:8080) and write new artifacts into the **Data Lake** (HTTP:9000).
6. **Monitoring Service:**
   * The Monitoring service (HTTP:9090) then pulls metrics from the registry and exposes them on a **Dashboard** (HTTP:3000), closing the loop on continuous model operations and observability.