

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

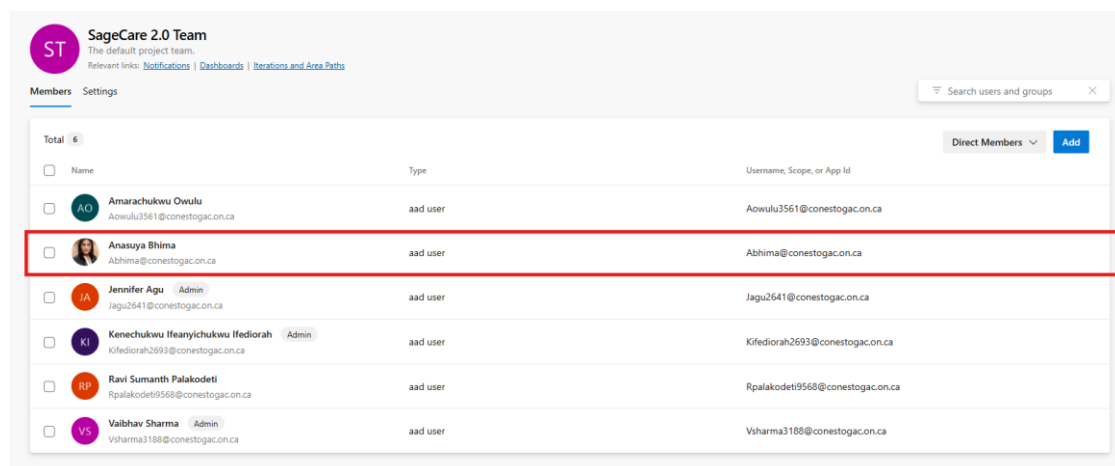


fig 1.1 Snapshot of team members including professor

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.

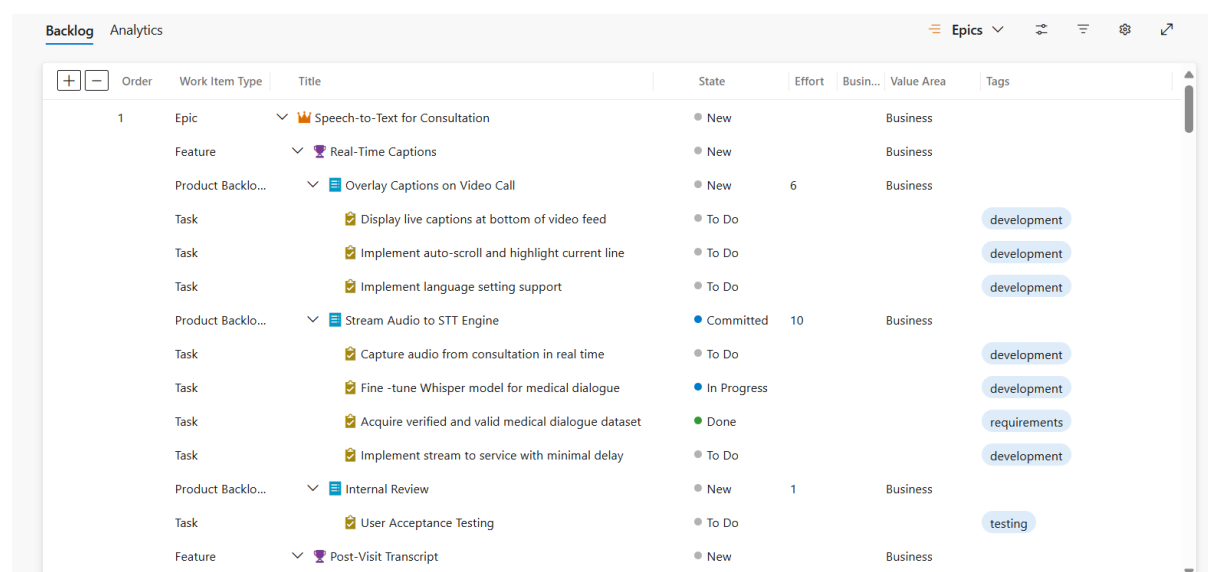


Fig 1.3 Snapshot of the groomed product backlog with epics. Features, PBIs and tasks for the sprint 1

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

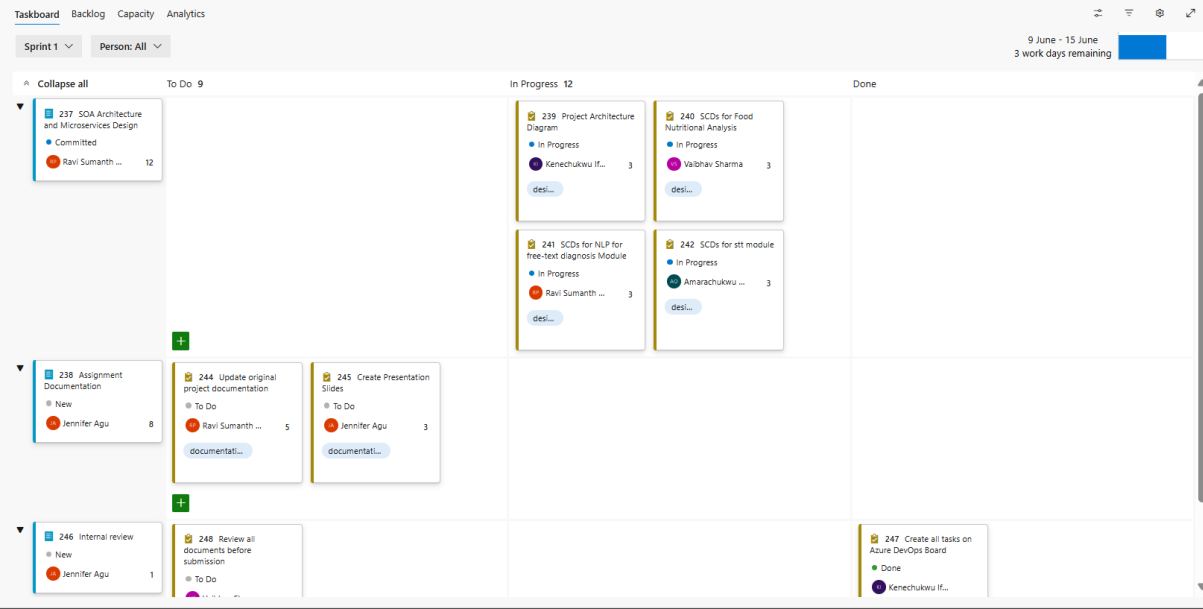
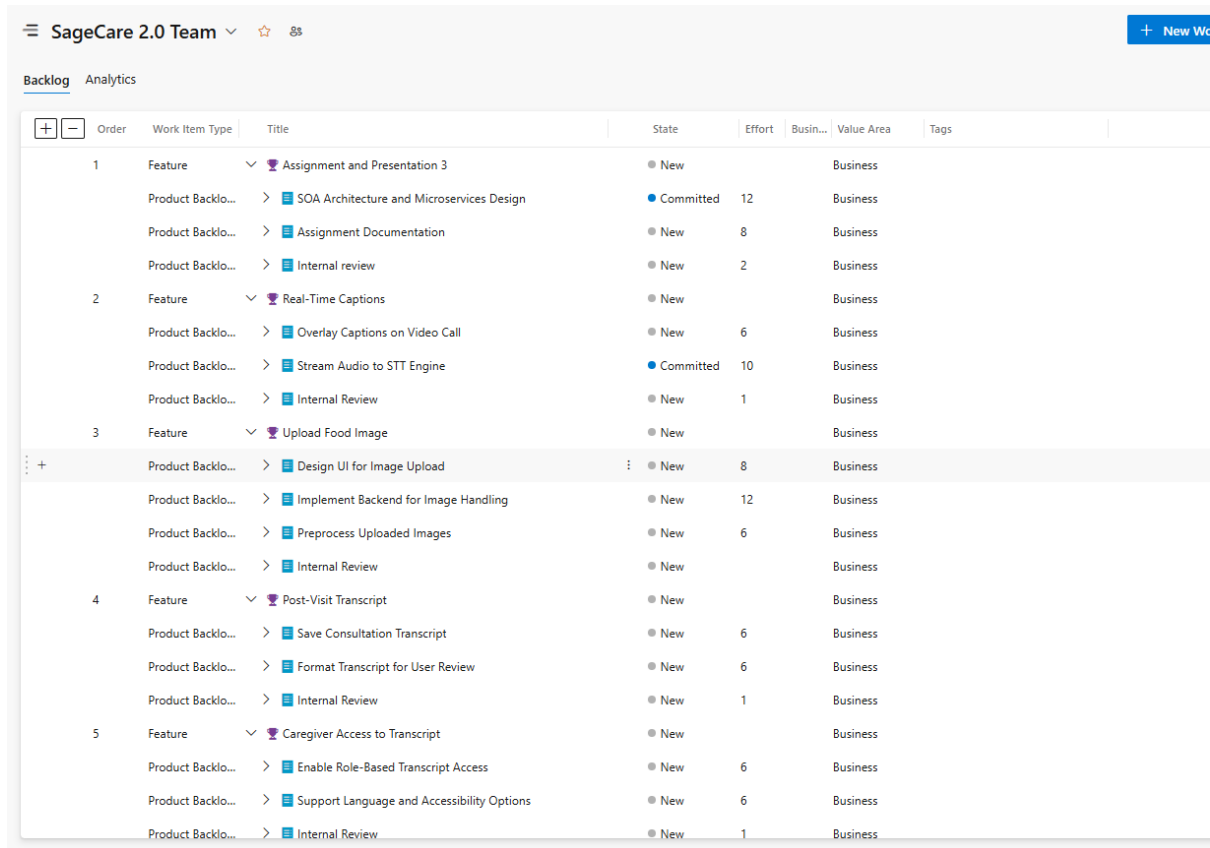


Fig 1.4 Snapshot of project board with tasks for sprint 1

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



Order	Work Item Type	Title	State	Effort	Busin...	Value Area	Tags
1	Feature	Assignment and Presentation 3	New			Business	
	Product Backlog...	SOA Architecture and Microservices Design	Committed	12		Business	
	Product Backlog...	Assignment Documentation	New	8		Business	
	Product Backlog...	Internal review	New	2		Business	
2	Feature	Real-Time Captions	New			Business	
	Product Backlog...	Overlay Captions on Video Call	New	6		Business	
	Product Backlog...	Stream Audio to STT Engine	Committed	10		Business	
	Product Backlog...	Internal Review	New	1		Business	
3	Feature	Upload Food Image	New			Business	
	Product Backlog...	Design UI for Image Upload	New	8		Business	
	Product Backlog...	Implement Backend for Image Handling	New	12		Business	
	Product Backlog...	Preprocess Uploaded Images	New	6		Business	
	Product Backlog...	Internal Review	New			Business	
4	Feature	Post-Visit Transcript	New			Business	
	Product Backlog...	Save Consultation Transcript	New	6		Business	
	Product Backlog...	Format Transcript for User Review	New	6		Business	
	Product Backlog...	Internal Review	New	1		Business	
5	Feature	Caregiver Access to Transcript	New			Business	
	Product Backlog...	Enable Role-Based Transcript Access	New	6		Business	
	Product Backlog...	Support Language and Accessibility Options	New	6		Business	
	Product Backlog...	Internal Review	New	1		Business	

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.

246 Internal review

JR Jennifer Agu

0 Comments Add Tag

StateNew

AreaSageCare 2.0

ReasonNew backlog item

IterationSageCare 2.0\Sprint 1

Description

Coordinate and follow up on all agile processes. Review all submission documents and tasks

Acceptance Criteria

- All documents are produced for submission on time
- All scrum artifacts are adhered to
- Consistent follow-up on tasks

Discussion

JR

Add a comment. Use # to link a work item, @ to mention a person, or ! to link a pull request.

switch to Markdown editor

Details

Priority2

Effort2

Business Value

Value areaBusiness

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.

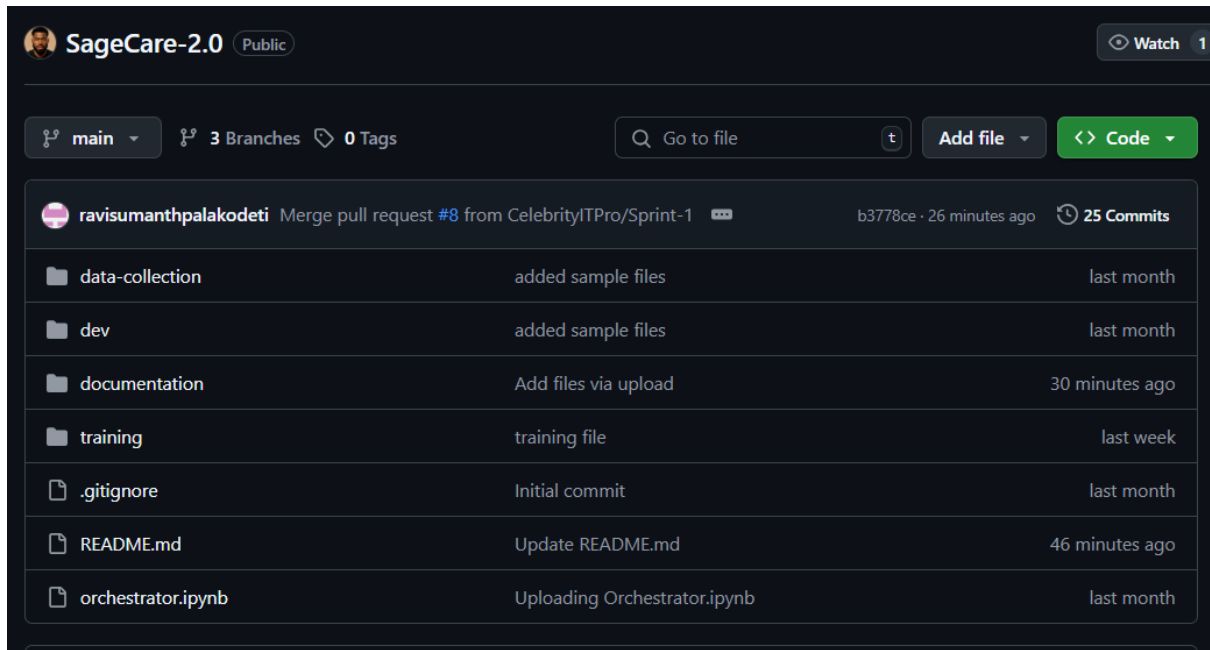


Fig 1.6 Snapshot of GitHub folder structure

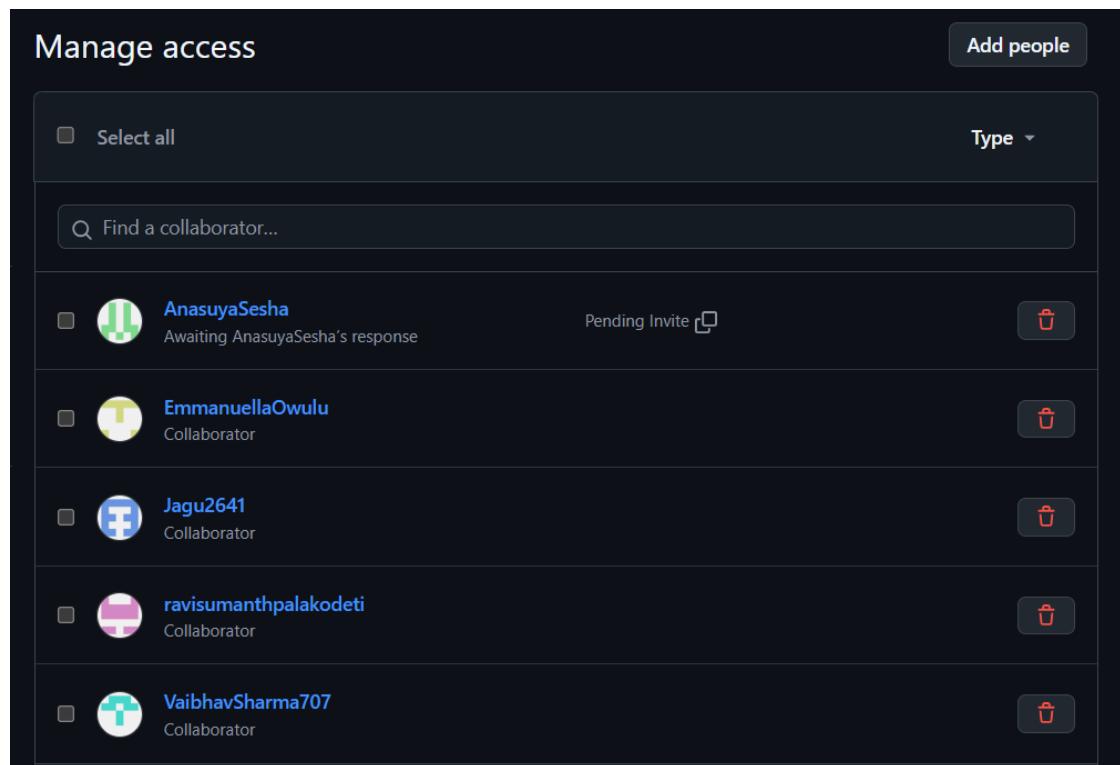


Fig 1.7 Snapshot of GitHub Repo access

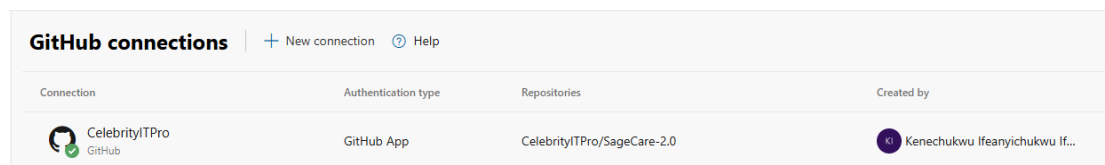


Fig 1.8 GitHub Repo integration with Azure DevOps board

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.

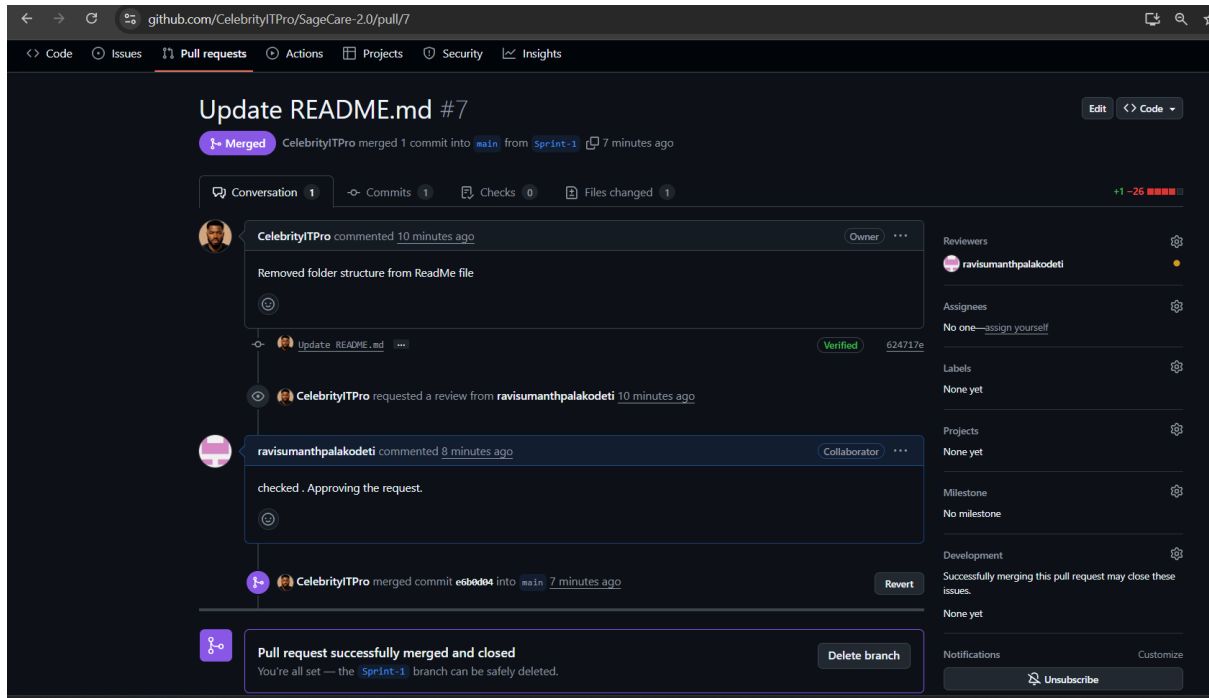


Fig 1.9 GitHub pull request issue and approval

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.

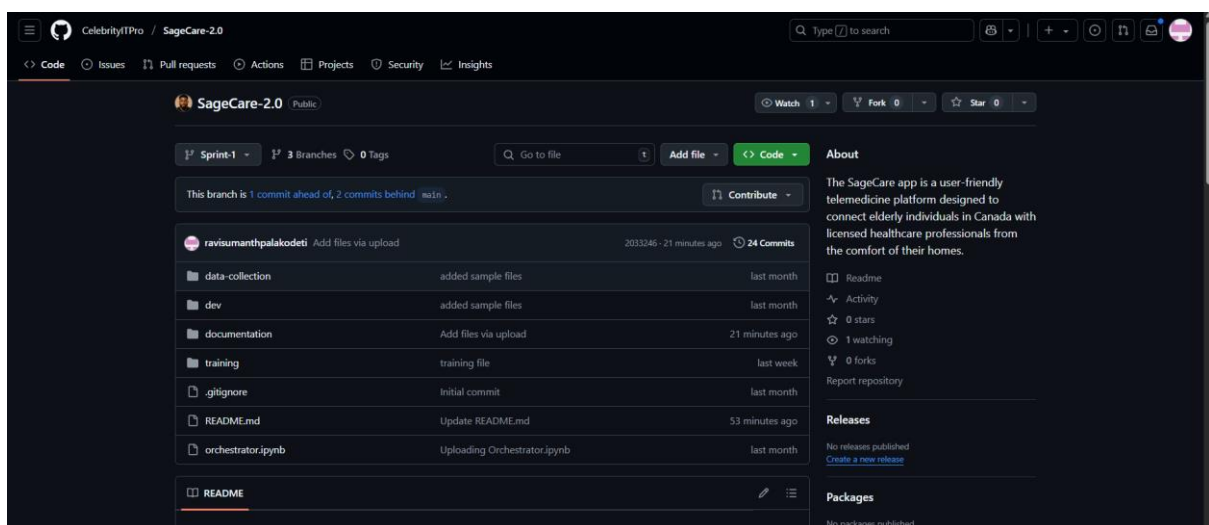


Fig 1.10 GitHub Sprint 1 branch

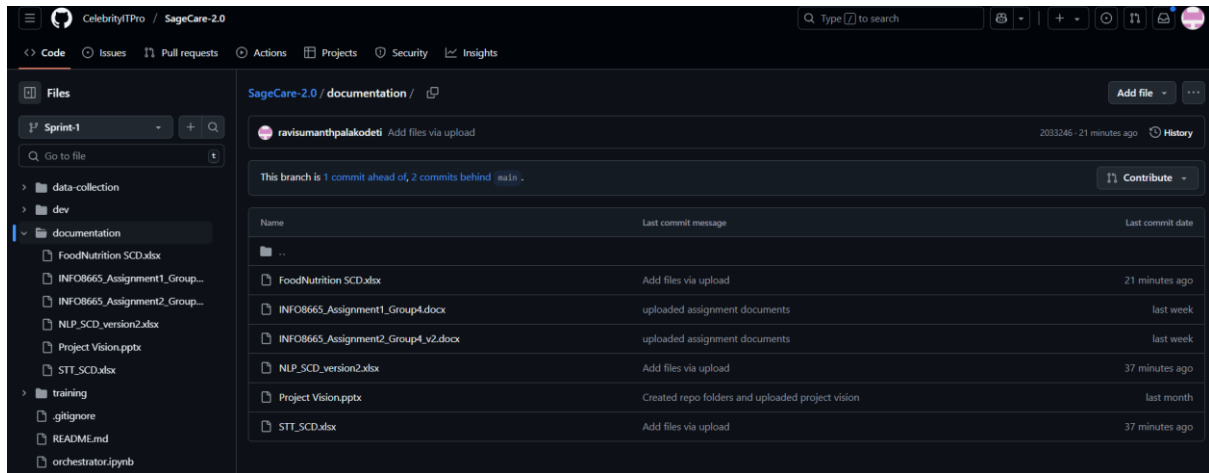


Fig 1.11 GitHub sprint 1 file updates

1.5 Team Roles and Responsibilities

Names	Roles	Responsibilities
Ifediorah Kenechukwu	Project Manager	<ul style="list-style-type: none"> - Break down tasks and ensures alignment with vision - Work on project submission documents - Works on ML model development
Agu Jennifer	Scrum Master	<ul style="list-style-type: none"> - Ensures team follows agile process and all tasks are updated accordingly - Works on ML model development - Manages Azure DevOps board
Palakodeti Ravi Sumanth	Developer	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - Designs the product frontend interface - Develops backend and frontend architectures - Works with APIs, databases, and UI.
Owulu Amarachukwu	Software Tester	<ul style="list-style-type: none"> - 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.

Priority List	Method	EndPoint	Description	Label	
1	POST	/api/v1/auth/signup	Register a new user account	signup-post	
2	POST	/api/v1/auth/login	Authenticate user and issue token	login-post	
3	POST	/api/v1/auth/logout	Invalidate refresh token and log out	logout-post	
4	GET	/api/v1/user/profile	Fetch user profile info	profile-get	
5	PUT	/api/v1/user/profile	Update user profile details	profile-put	
6	GET	/api/v1/classify	Classify uploaded meal image	classify-get	
7	GET	/api/v1/foods/{foodId}	Fetches food nutrition content	food-nutrient-get	
8	GET	/api/v1/recommendations/daily?date={ISO}	Returns personalised meals advice	recommendation-get	
9	PUT	/api/v1/classify/{imageId}	Update classification result manually	classify-put	
10	POST	/api/v1/images/analyze	Analyze uploaded meal photo returns	analyze-post	

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.

POST	https://api.sagecare.ca/api/v1/auth/signup	Register a new user account
Parameters		
Name	Description	
email	User email	
password	User password	
age	User age	
gender	User gender	
Responses		
Code	Description	
201	User created	
	Model	Example
	<pre>{ "id": integer, "email": string, "age": integer, "gender": string, "healthConditions": array }</pre>	<pre>{ "id": 123, "email": "alice@example.com", "age": 35, "gender": "female", "healthConditions": ["diabetes"] }</pre>
	need details from Patrick	need details from Patrick
400	Validation error	
409	Email exists	

< >

Labels

signup-post

login-post

logout-post

profile-get

profile-put

classify-get

food-nut

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.

Priority	End Point	Method	Params	Body	Result
1	/api/v1/auth/register	POST		{email, password}	{user_id, message}
2	/api/v1/auth/login	POST		{email, password}	{access_token, token_type, expires_in}
3	/api/v1/symptoms/submit	POST		{user_id, description}	{symptom_id, message}
4	/api/v1/symptoms/{symptom_id}	GET	symptom_id		{symptom_id, user_id, description, submitted_at}
5	/api/v1/preprocess/clean	POST		{symptom_id, raw_text}	{symptom_id, clean_text}
6	/api/v1/diagnosis/predict	POST		{symptom_id, clean_text}	{diagnosis_id, predicted_diagnoses}
7	/api/v1/mapping/specialties	POST		{diagnosis_id, predicted_diagnoses}	{diagnosis_id, specialties}
8	/api/v1/referrals/create	POST		{user_id, diagnosis_id, specialty}	{referral_id, status, message}
9	/api/v1/referrals/{user_id}	GET	user_id		[[referral_id, specialty, status, created_at]]
10	/api/v1/history/analyses/{user_id}	GET	user_id		[[analysis_id, symptom_id, raw_text, predicted_diagnoses, specialties, referral_status, timestamp]]

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

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

1. Register New User		
Method	End Point	Description
POST	/api/v1/auth/register	Register new User
Parameters		
Name	Description	
email	String	
password	String, min 8 chars	
Request		
	Model	Example
	{	{
	"email": "string, email format",	"email": "john.doe123@email.com"
	"password": "string, min 8 chars"	"password": "Password@123"
	}	}
Responses		
Code	Description	
201	Created	
	{	{
	"user_id": "uuid",	"user_id": "Johndoe123",
	"message": "User registered successfully."	"message": "User registered successfully."
	}	}
400	Bad Request : Missing Fields, invalid email or weak password	
	{ "error": "Invalid email or password does not meet policy." }	
409	Conflict : Email Already in use	
	{ "error": "Email already registered." }	

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.

EndPoint	Description	Label
/api/v1/auth/signup	Register a new user account	signup-post
/api/v1/auth/login	Authenticate user and issue token	login-post
/api/v1/auth/logout	Invalidate refresh token and log out	logout-post
/api/v1/user/profile	Fetch user profile info	profile-get
/api/v1/user/profile	Update user profile details	profile-put
/api/v1/doctor/search	Search doctors by specialty, name or availability	doctor-search-get
/api/v1/appointment/schedule	Schedule a new consultation appointment	Appointment-schedule-post
/api/v1/appointment/user	Retrieve user's upcoming and past appointments	appointment-user-get
/api/v1/appointment/{appointmentId}/cancel	Cancel a scheduled appointment	appointment-cancel-delete
/api/v1/meeting/join	Join a video consultation session	meeting-join-post
/api/v1/meeting/consent	Submit patient consent for call recording/transcription	consent-post
/api/v1/record/start	Start audio recording of the consultation	record-start-post
/api/v1/record/upload	Upload audio recording to server	record-upload-post
/api/v1/transcription/run	Transcribe uploaded audio file	transcription-run-post
/api/v1/transcription/{appointmentId}	Retrieve transcript by appointment ID	transcription-get
/api/v1/transcription/feedback	Submit feedback on transcript accuracy	transcription-feedback-post
/api/v1/insights/{appointmentId}	View nutritional insights based on transcript	insights-get
/api/v1/alerts	Trigger alerts for critical health patterns	alerts-post
/api/v1/recommendation/generate	Generate meal or health suggestions based on insights	recommendation-post
/api/v1/recommendation/customize	Customize recommendations based on user profile	recommendation-customize-post
/api/v1/history/nutrition	Retrieve nutrition history over time	history-nutrition-get
/api/v1/history/transcripts	View transcript history	history-transcript-get
/api/v1/feedback/general	Submit general feedback on consultation or app	feedback-general-post
/api/v1/feedback/user	Get feedback submitted by user	feedback-user-get
/api/v1/support/faqs	Fetch frequently asked questions	support-faqs-get

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

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

POST	http://api/v1/auth/signup	Register a new user account
Parameters		
Name	Description	
name	Full name of the user	
email	User's email address	
password	Password for the account	
role	Role (e.g., "patient", "doctor")	
	200	Success
	Model	Example
	{ user_id: string, email: string, message: string }	{ user_id: "u123", email: "john@example.com", message: "User registered successfully" }
	400	Missing or invalid fields
	401	Email already exists
	500	Server error during registration

Fig 1.18 Register Service RESTful API call

1.6.2 Application Architecture

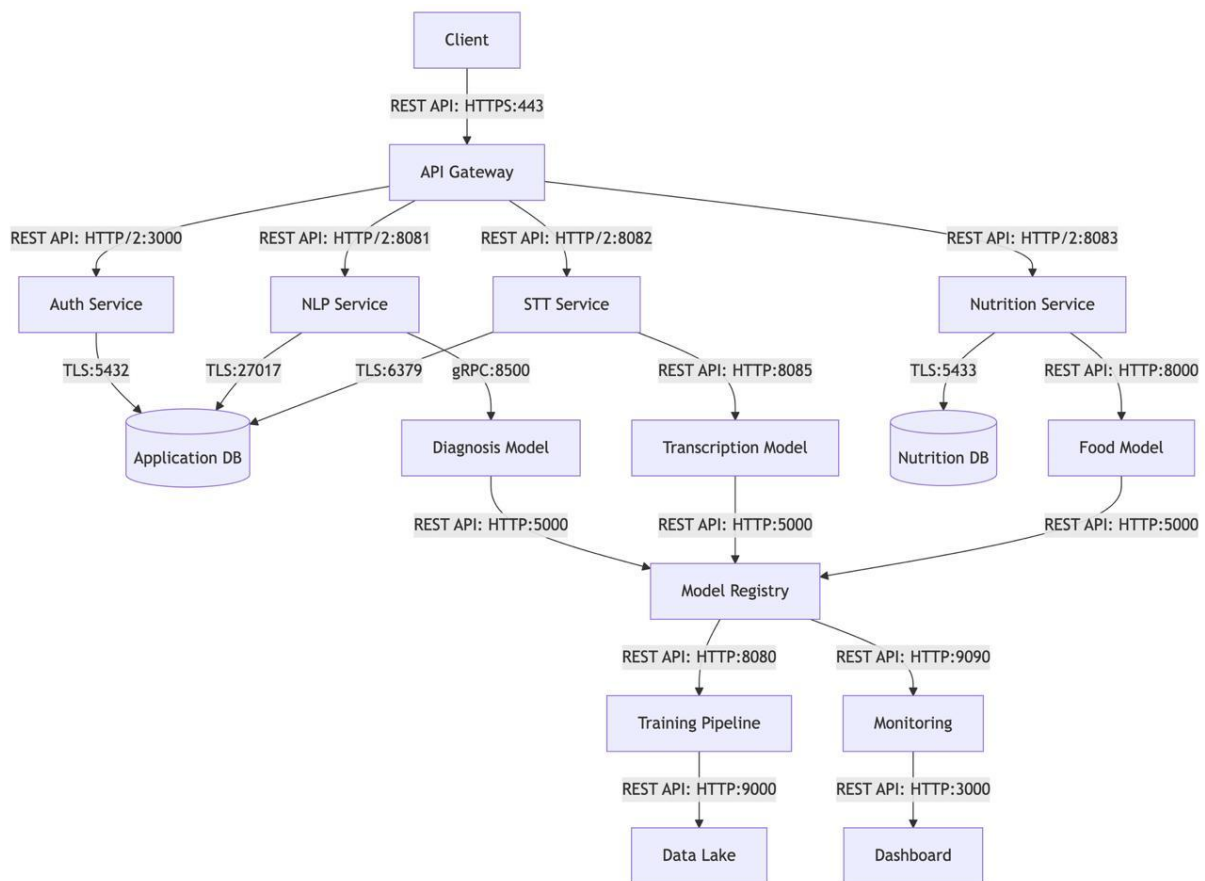


Fig 1.19 Architecture diagram

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

2. 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.

3. 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.

4. The “Speech to Text Services”

- This implements speech-to-text features, feeding transcribed audio into the application database for downstream processing.

5. **Nutrition Service:**

- It runs on its own endpoint but registers its inference model with the central Model Registry.

6. **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).

7. **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.