**Sri Lanka Institute of Information**

**Technology**

A logo with a lion and a lion on it

Description automatically generated

**Project Deployment Report**

**Project Title: AI-enabled-Intelligent-Assistant-to-Improve-Reading-and-Comprehension-Skills-in-English**

**Group Members (Group ID- RP24-25J-027)**

|  |  |
| --- | --- |
| IT21173790 | Sooriyaarachchi .M.D.A |
| IT21158322 | Senanayake W.G.B |
| IT21118340 | Kumarathunga S.A.D.S |
| IT21182396 | Ranaweera A.P |

**1.Project Overview**

This report documents the development, deployment, and testing of our AI-powered English learning platform. It follows the Software Development Life Cycle (SDLC) and utilizes a microservices architecture to ensure modularity, scalability, and maintainability. The frontend is developed in React + Vite, while backend services are containerized and deployed on AWS ECS using CI/CD pipelines.

**Key Metrics:**

* Deployment Success Rate: 100%
* Downtime: 0 hours during deployment
* Uptime: 99.9% post-deployment
* CI/CD Status: Successful Docker image builds and GitHub Actions workflows

**2. Deployment Objectives**

**Goals:**

* Deliver a scalable, modular web application for English learning.
* Ensure each microservice operates independently and reliably.
* Deploy both frontend and backend components using modern DevOps practices.

**Scope:**

* Frontend built with React and Vite.
* 5 microservices for backend deployed on AWS ECS.
* Frontend deployed on Netlify.
* CI/CD with Docker and GitHub.

**3. Software Specifications**

**3.1 Functional Requirements**

* User registration and login
* AI-based vocabulary training
* Dynamic comprehension tests (basic and advanced)
* Speech error detection
* Real-time feedback and user progress tracking
* It should be able to analyze student-written input and predict their CEFR vocabulary proficiency level.
* It should use natural language processing to extract meaningful features like lexical richness and syntactic complexity.
* It should be able to generate vocabulary games (e.g., fill-in-the-blank, association games) based on user level and preferences.
* It should include a chatbot assistant capable of explaining word meanings,

**3.2 Non-Functional Requirements**

* Scalability via container orchestration
* High availability with auto-scaling ECS services
* Fast response via Vite + CDN (Netlify)
* Secure communication via HTTPS and JWT tokens

**4. Pre-Deployment Activities**

**Environment Preparation:**

* Set up AWS ECS clusters and services
* Created S3 bucket for assets
* Configured Netlify deployment from GitHub

**Testing:**

* Unit testing of each backend microservice
* End-to-end integration testing
* User Acceptance Testing (UAT) via Google Forms

**Backup Plan:**

* Codebase backed up via GitHub repositories
* AWS ECS configurations exported
* Daily snapshot backups for ECS volumes

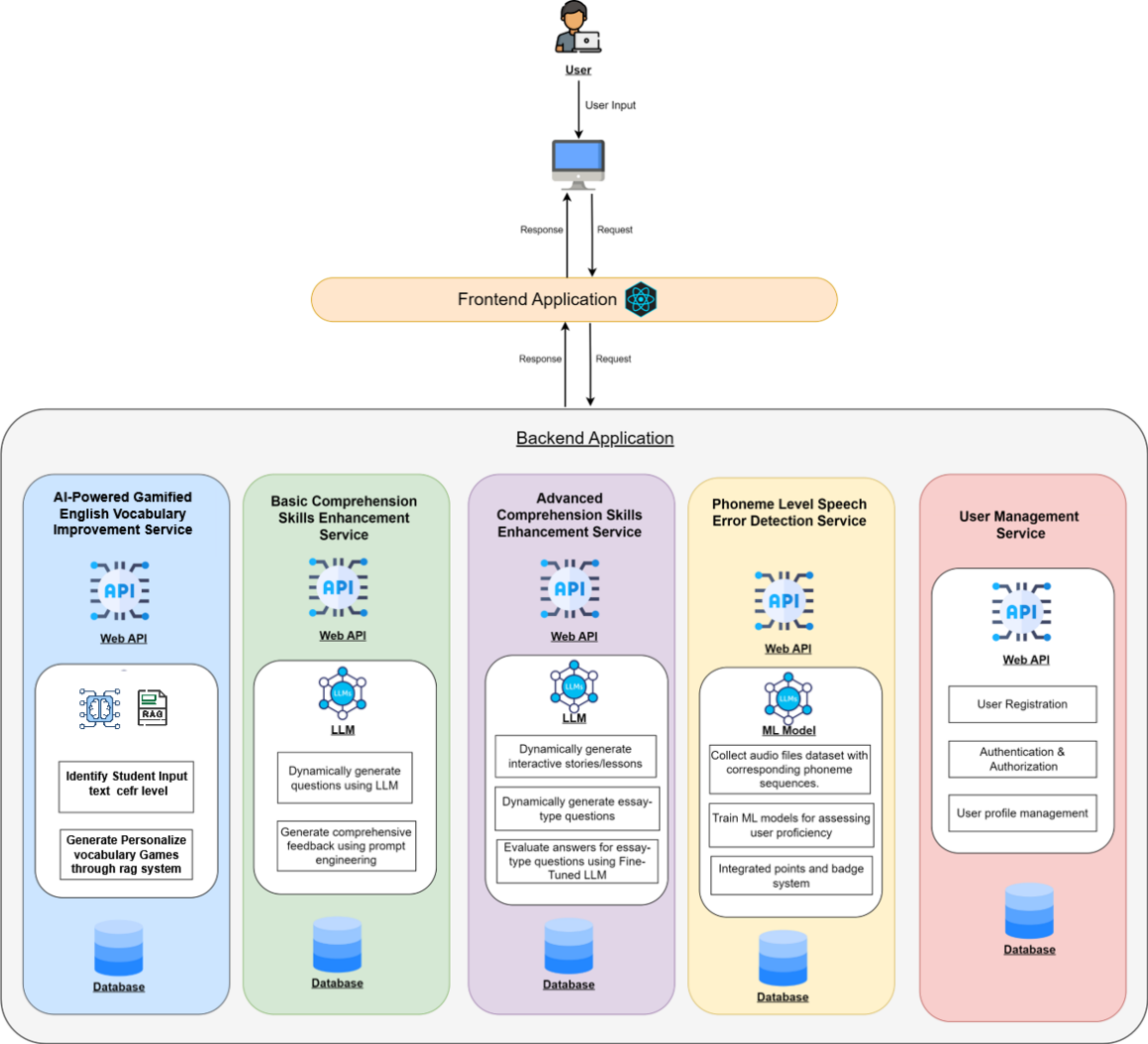
**5. Architecture Overview**

Based on your diagram, the system includes:

**Frontend**

* **React + Vite App** deployed on **Netlify**
* Handles user input and routes it to relevant backend services via API calls

**Backend (5 Microservices)**



|  |  |
| --- | --- |
| **Service Name** | **Description** |
| AI-Powered Gamified Vocabulary Improvement | Uses AI to identify and gamify vocabulary gaps |
| Basic Comprehension Skills Enhancement | Dynamically generates questions using LLMs |
| Advanced Comprehension Skills Enhancement | Context-based dynamic comprehension, integrated with gamification |
| Phoneme-Level Speech Error Detection | Detects speech errors using ML and provides audio feedback |
| User Management Service | Handles user registration, login, and profile management |

Each service:

* Exposes its own **REST API**
* Has its own **database**
* Is independently deployed via **Docker containers to AWS ECS (Fargate)**

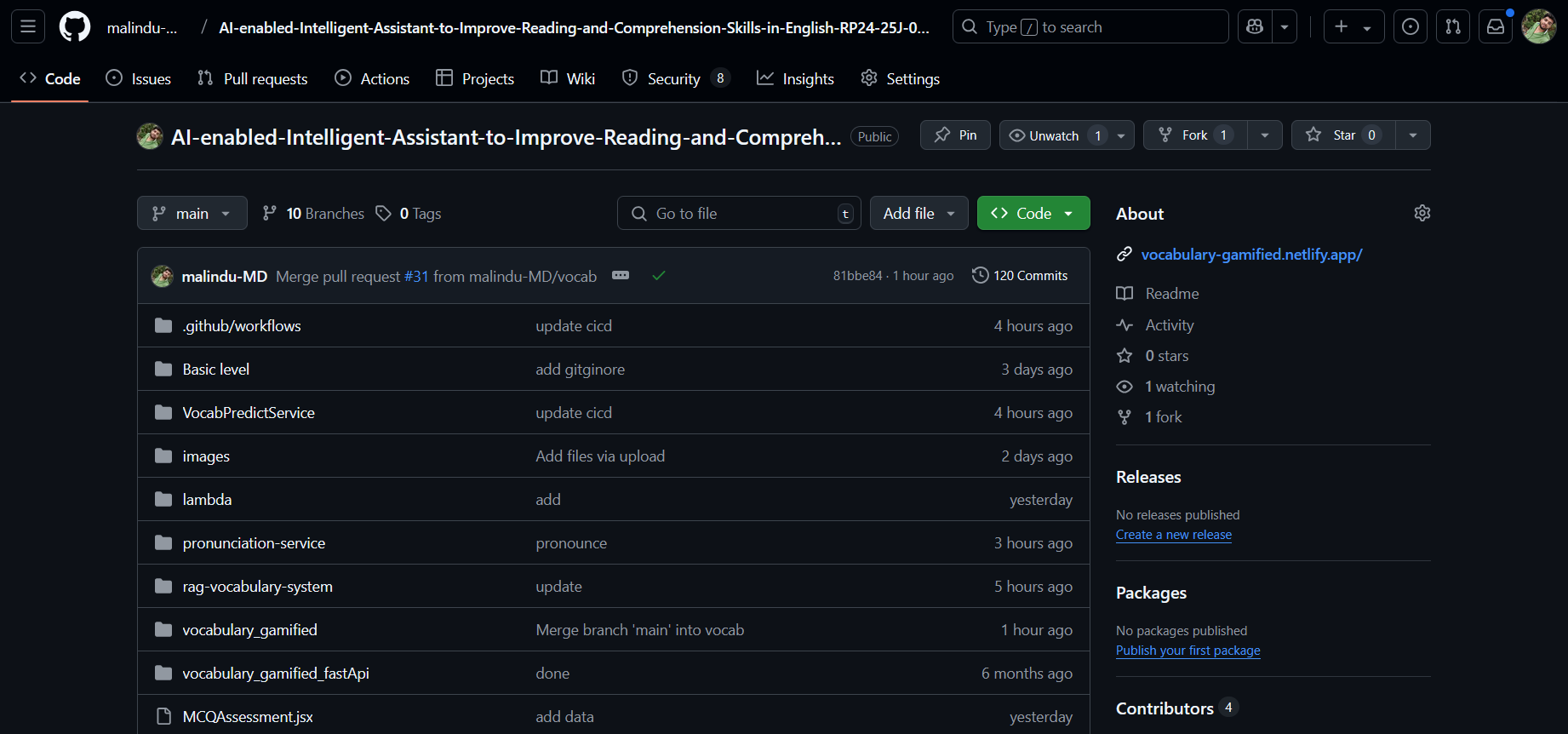
**6.Tools and Technologies**

|  |  |  |
| --- | --- | --- |
| **Category** | **Tool / Platform** | **Version / Details** |
| Frontend | React + Vite | React 18 / Vite 5 |
| Backend Services | Fast API | Service-specific stack |
| Database | PostgreSQL/Firebase /Pinecone Vector DB | Dedicated DB per service |
| Containerization | Docker / Docker Compose | Docker 24.0 |
| CI/CD | GitHub Actions | Automated test/build/deploy |
| Hosting Platform | AWS ECS (Fargate) | Container orchestration |
| API Gateway | AWS API Gateway | Unified entry point for services |
| Static Hosting | Netlify | Frontend build and deploy |
| ML/AI Integration | Gemini AI / Speech Models/AWS /Higgin face/ Transformer Models / Lang Chain /NLP models/Embedding Models | Used For all Modules |

**7. Deployment Process**

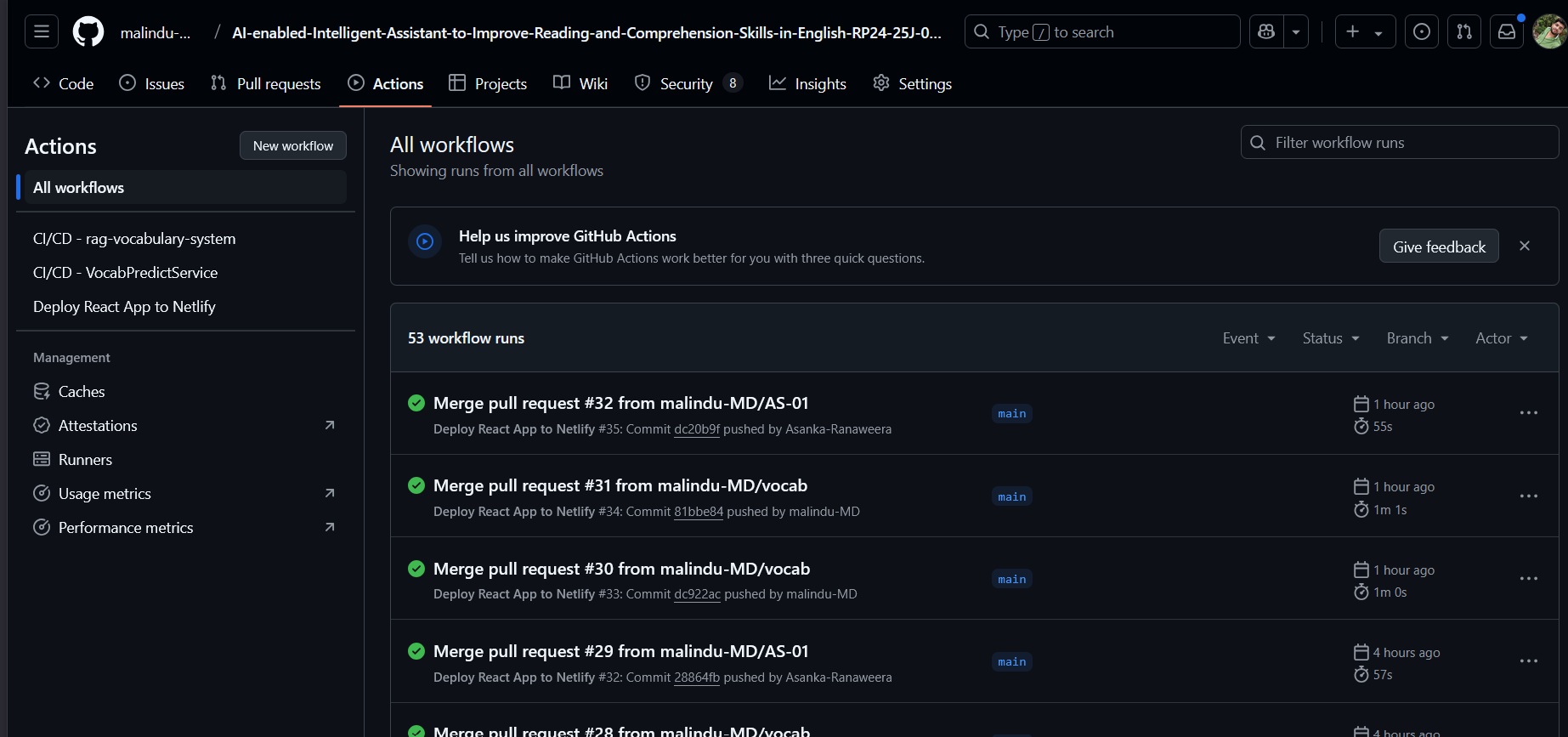
7.1 CI/CD via GitHub Actions

* Each microservice is in a GitHub repository

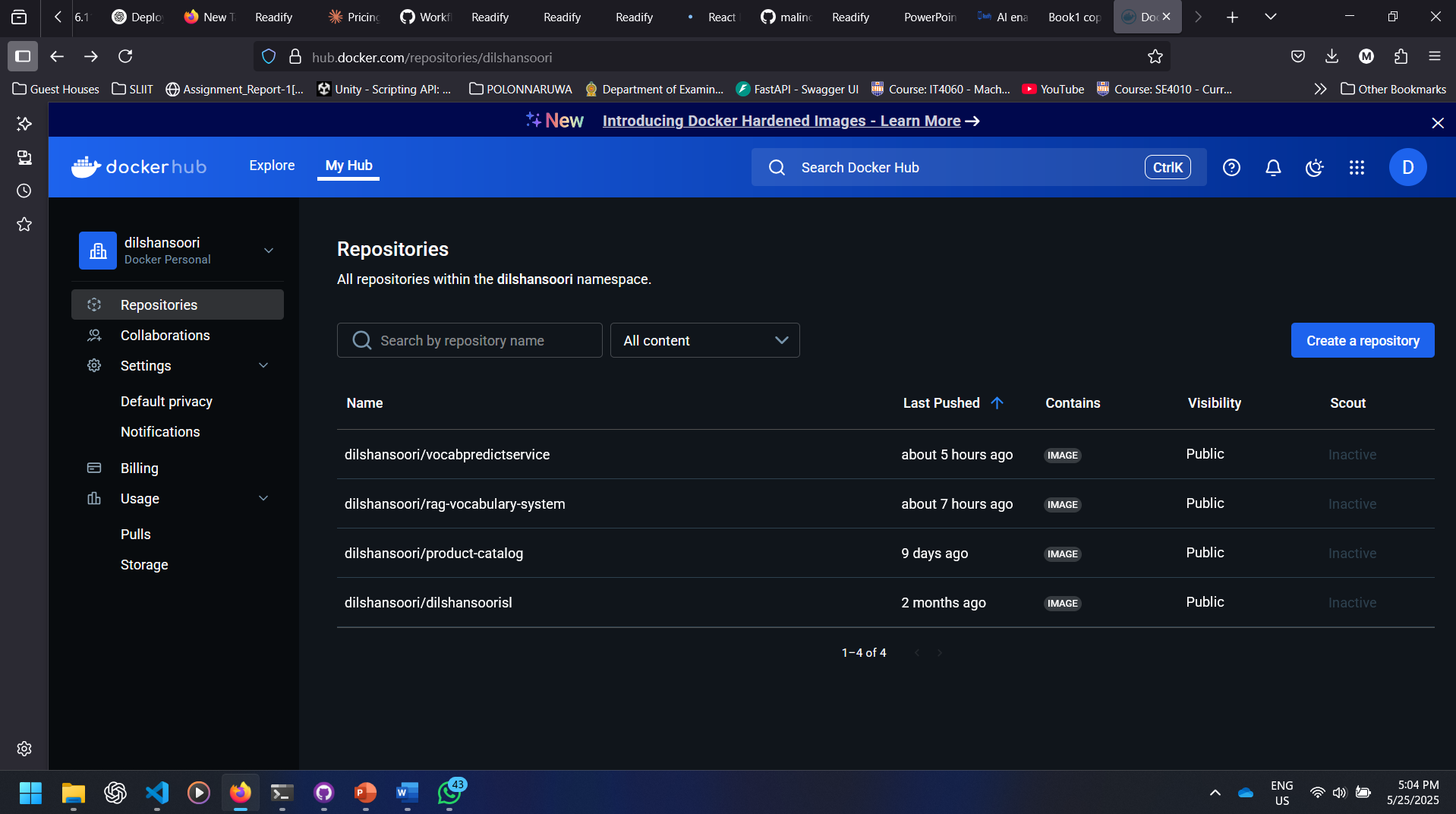


**Figure** Error! No text of specified style in document.**.1 GitHub repository**

* CI triggers on push/pull\_request:
  + Run tests
  + Build Docker image
  + Push to Docker Hub



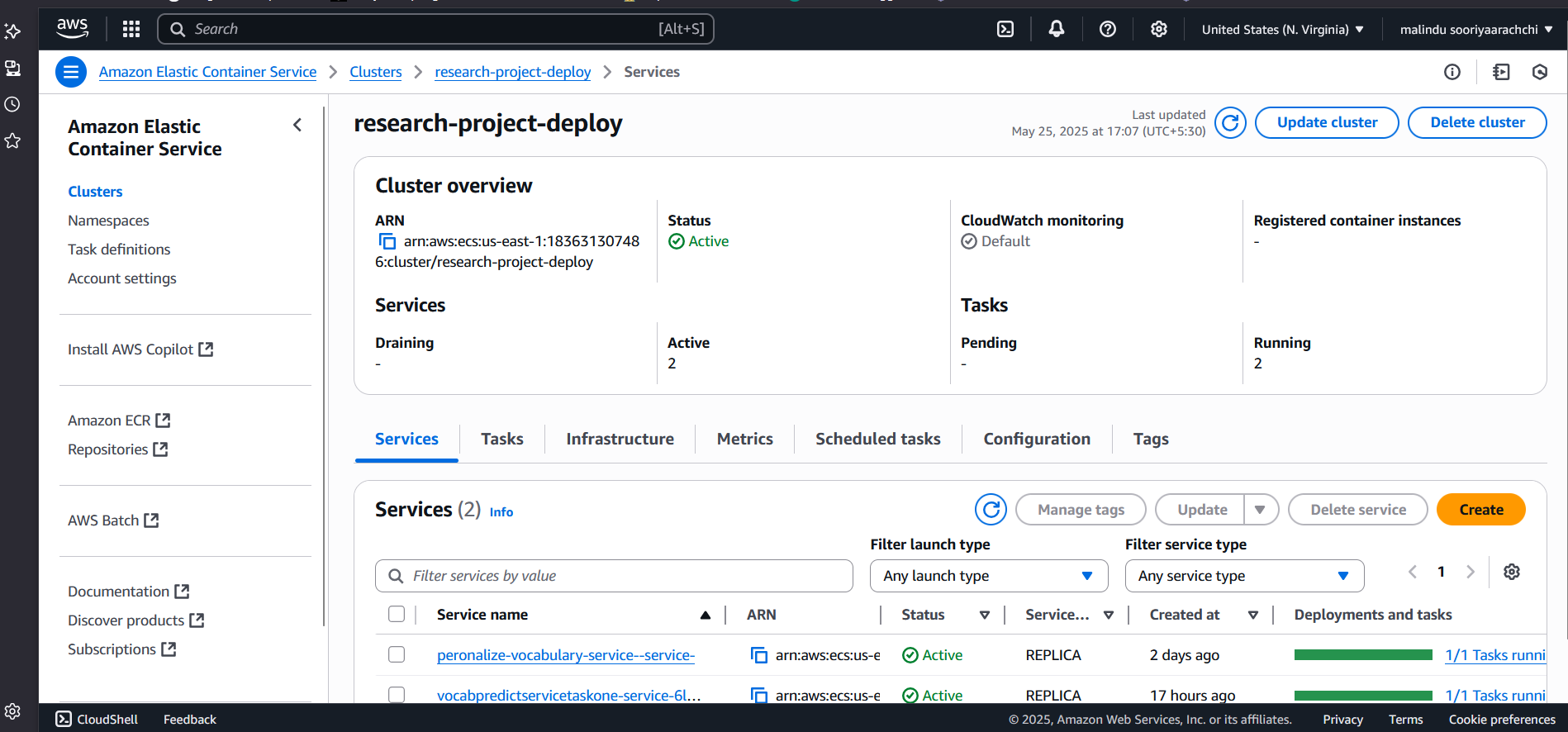
**Figure 2 GitHub Actions log showing successful build**



**Figure 3 Docker HUB showing Docker images pushed**

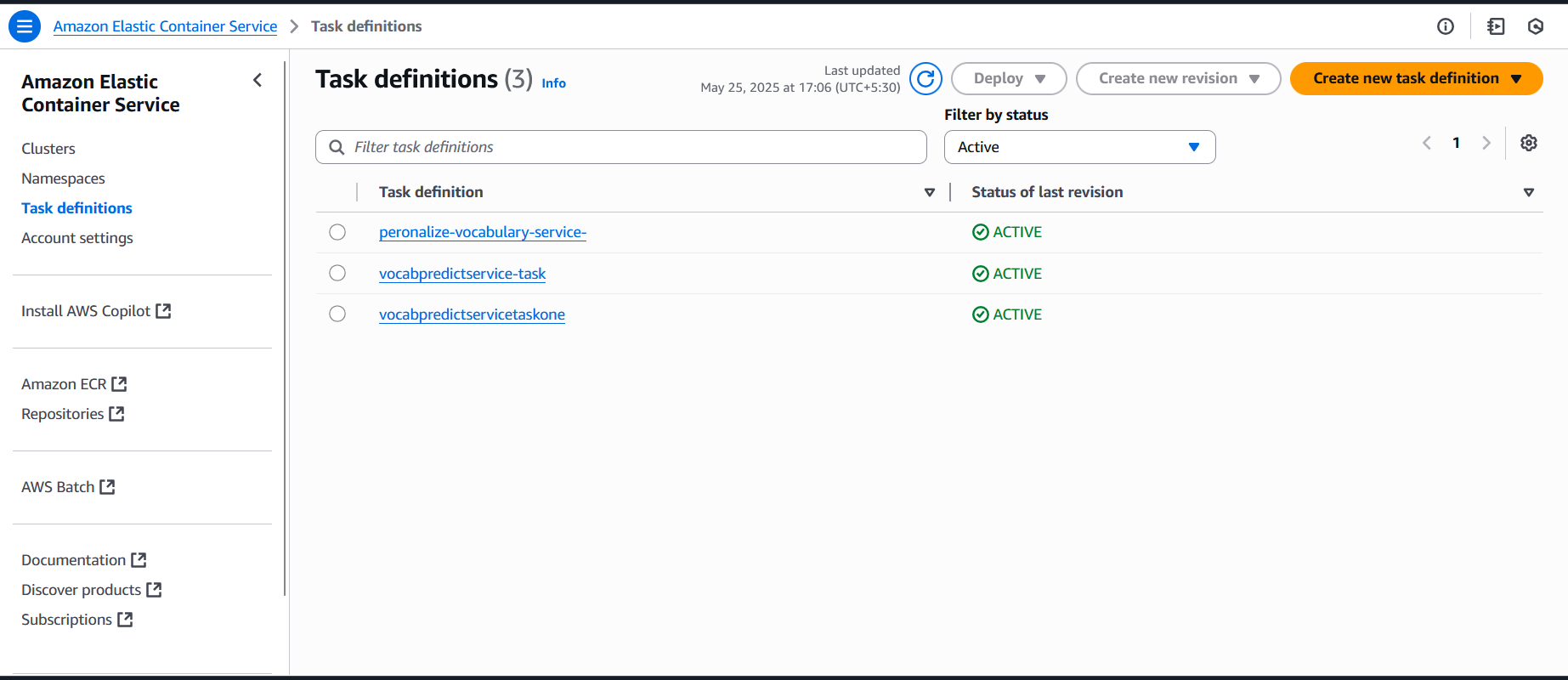
7.2 AWS ECS (Fargate) Deployment

* Services deployed independently to ECS



**Figure 4 AWS ECS dashboard with running services**

* ECS Task Definitions used for auto-scaling

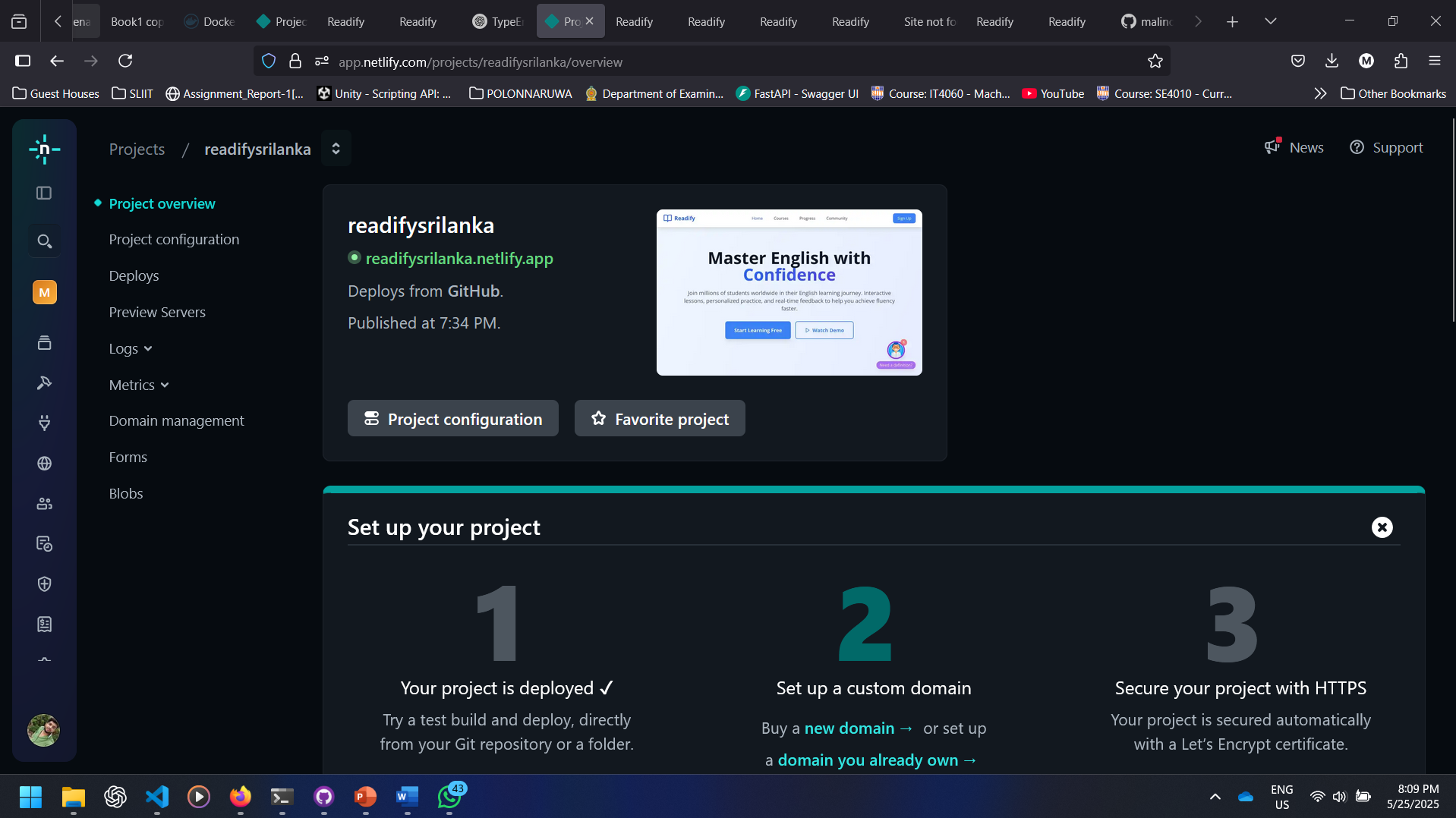


**Figure 5 AWS ECS Task Definition**

* Load balancer + service discovery via AWS ALB
* Logs and metrics via CloudWatch

7.3 Frontend on Netlify

* Connected to GitHub repo
* Auto-deploy on main push
* Vite build generates production bundle
* Hosted globally with HTTPS



**Figure 6 Netlify Deployed Dashboard**



**Figure 7 Frontend UI Preview**

**8. Cloud Configuration Details**

|  |  |  |
| --- | --- | --- |
| **Resource** | **Platform** | **Details** |
| Frontend | Netlify | HTTPS, Auto-build on push |
| Backend Services | AWS ECS | Fargate tasks, 0.5 vCPU, 3GB RAM |
| Container Registry | Docker/Docker Hub | One repo per microservice |
| API Gateway | AWS Gateway | Routes API requests |
| Databases | PostgreSQL (RDS), Pinecone DB, Firebase | One per service, private subnet |
| CI/CD | GitHub Actions | Auto-build, test, deploy |
| Domain & DNS | Route 53 / Netlify | SSL, custom domain |

**9. Testing**

**9.3 Test Case Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Component | Description | Status |
| TC01 | User Login API | Login with correct credentials | pass |
| TC02 | Speech Detection API | Upload audio, get feedback | pass |
| TC03 | Vocab Service | Correctly worked | pass |
| TC04 | Comprehension Generator | Generate questions from passage | pass |
| TC05 | Frontend Rendering | User sees dashboard | pass |

* Test Case for Vocab Service

Table 3:1 Test Case to CEFR Classification from Input Text

|  |  |
| --- | --- |
| **Test Case ID** | **TC-01** |
| **Scenario** | Verify CEFR level prediction from input paragraph |
| **Input** | “The scientific community has debated the ethical use of AI.” |
| **Expected Output** | CEFR Level: C1 or C2 |
| **Actual Output** | CEFR Level: C1 |
| **Status** | Pass |

Table 3:2 Test Case to RAG Game Generation Based on Topic and Level

|  |  |
| --- | --- |
| **Test Case ID** | **TC-02** |
| **Scenario** | Verify quiz generation using RAG system with input topic |
| **Input** | Topic: “Technology”, CEFR: B1 |
| **Expected Output** | Game JSON with MCQs and contextual vocabulary |
| **Actual Output** | 10 MCQs returned with tech-related content |
| **Status** | Pass |

Table 3:3 Test Case to Chatbot Vocabulary Response

|  |  |
| --- | --- |
| **Test Case ID** | **TC-03** |
| **Scenario** | User queries chatbot for word explanation |
| **Input** | User message: “Define cryptocurrency” |
| **Expected Output** | Definition + example + related terms |
| **Actual Output** | Correct response generated |
| **Status** | Pass |

Table 3:4 Test Case to Game UI Rendering

|  |  |
| --- | --- |
| **Test Case ID** | **TC-04** |
| **Scenario** | Verify quiz renders correctly on UI |
| **Input** | Injected JSON with MCQ and options |
| **Expected Output** | 1 question, 4 options, selection feedback |
| **Actual Output** | Accurate rendering and scoring |
| **Status** | Pass |

Table 3:5 Test Case to Preference-Based Game Filtering

|  |  |
| --- | --- |
| **Test Case ID** | **TC-05** |
| **Scenario** | Check game generation respects topic preference |
| **Input** | Preference: “Sports”, Level: A2 |
| **Expected Output** | Questions about sports vocabulary |
| **Actual Output** | FITB and MCQs on “goalkeeper”, “match”, etc. |
| **Status** | Pass |

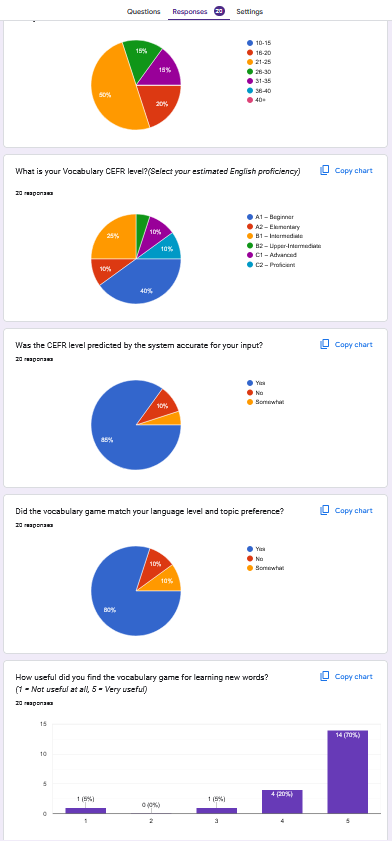
Table 3:6 Test Case to Fill-in-the-Blank Question Generation

|  |  |
| --- | --- |
| **Test Case ID** | **TC-06** |
| **Scenario** | Check LLM generation of FITB questions |
| **Input** | Word: “firewall”, Level: B1 |
| **Expected Output** | FITB JSON with 4 options |
| **Actual Output** | Game returned with structured JSON |
| **Status** | Pass |

Table 3:7 Test Case to Hint Prompt Generation Accuracy

|  |  |
| --- | --- |
| **Test Case ID** | **TC-07** |
| **Scenario** | Generate a hint for word without revealing it |
| **Input** | Word: “encryption” |
| **Expected Output** | Concise, indirect definition |
| **Actual Output** | “Used to protect information using codes.” |
| **Status** | Pass |

* Google Forms for UAT to validate frontend usability, content relevance, and visual design with real user feedback



**10. Issues and Resolutions**

Encountered Issues:

* Netlify failed due to environment variable mismatch
* One ECS task failed due to port conflict

Resolutions:

* Updated Netlify env configuration
* Adjusted ECS task definition port settings

Impact Analysis:

* No downtime experienced by users
* Recovery time: < 30 minutes

**11. Conclusion**

Our system is a robust, AI-enhanced English learning platform built on microservices principles. It supports secure, scalable, and independent deployments of features and uses modern DevOps and cloud tools for reliability. Each component is tested and monitored, ensuring smooth performance across services.

**GitHub Repositories**: [link](https://github.com/malindu-MD/AI-enabled-Intelligent-Assistant-to-Improve-Reading-and-Comprehension-Skills-in-English-RP24-25J-027)

**Demo Link**: <https://readifysrilanka.netlify.app/>