PROJECT SUMMARY

Project Title: Attack the Virus

CS-442, Fall 2024

Group 4: Apoorv Lodhi, Niyati Malik, Ragini Kalvade, Ryder Douglas

Project Overview

"Attack the Virus" (ATV) is an educational web application designed to increase public awareness about vaccines, viruses, and the immune system. The app targets students, educators, and individuals interested in learning about public health. By combining engaging gameplay with informative content, ATV offers a fun and interactive way to educate users about the importance of vaccines, disease prevention, and immune system functions.

Deliverables

The core deliverables of ATV include:

1. Interactive Quiz: A multiple-choice game that educates users about vaccines, the immune system, and disease prevention, with immediate feedback and informative "Info Cards."

2. Mini-Games: Virus-themed games like Virus Breaker, Virus Sweeper, and Virus Panic that combine education and entertainment.

3. Clinic Finder: A tool powered by Google Maps to help users find nearby clinics, doctors, and hospitals.

4. AI Chatbot "Vacciwiz": A chatbot that provides real-time information about viruses, vaccines, and health-related questions.

These features work together to provide an engaging and informative experience.

Testing

The testing for the "Attack the Virus" project was conducted using JUnit and Mockito frameworks, focusing on three key areas: Controller Layer, Service Layer, and Frontend Components. In the Controller Layer tests, all API requests were validated for correct handling, with 7 test cases for AgentController, 3 for LeaderboardController, and 9 for QuizController, all passing. The Service Layer testing involved ensuring proper data processing and repository interactions, with 8 test cases for AgentService, 5 for LeaderboardService, and 6 for QuizService, all passing as expected. For Frontend Testing, Angular components like OtherGamesComponent, LeaderboardComponent, and QuizComponent were rendered with mock data and user interactions, verifying functionality and UI rendering with 8 test cases, all passing. Regression Testing was performed on critical components, confirming that no new issues emerged. The overall testing confirmed the project's robustness, with all test cases passing successfully.

Inspection

The inspections for the "Attack the Virus" app focused on five key areas. The map functionality was tested for accurate location tracking and nearby clinic display, with results verified against Google Maps. Quiz questions were reviewed for clarity, with usability testing done to ensure ease of understanding. Game rules were checked for simplicity, confirmed through testing with players to ensure clarity. Player inputs and progress were tested by entering valid and invalid data, ensuring progress was saved correctly across devices. Image loading was tested on various networks, ensuring placeholder images appeared on slower connections, and no missing image errors were found. All inspections were successful, ensuring smooth functionality across the app.

Recommendations and Conclusions

The "Attack the Virus" app passed testing and inspection for key features, including map functionality, quiz questions, game rules, and player input handling. All issues were addressed and retested during follow-up sprints. With critical functionalities validated, the application is ready for deployment.

Project Issues

The "Attack the Virus" project faced some uncertainties that could affect the final product, like issues with the clinic finder, relying on external APIs for disease and vaccine updates, and making sure the app works well on all devices. These problems were identified and will be looked at in future updates. Some features, like virus simulation, better avatar customization, and adjusting quiz difficulty, were put on hold for now because they require more resources and time to develop. The quiz feature will be a priority for the next update.

The project worked well with tools like Jira Kanban boards and daily meetings to keep everyone on track. However, some challenges came up, like bottlenecks in pair programming, short one-week sprints, and adding extra features mid-sprint. In the future, using longer sprints, adding automated testing, and focusing on managing the scope of the project will help improve efficiency and reduce rushed work.