**PBT205—Project-based Learning Studio**

**Assessment 3 - Prototype extension and report**

**Torrens University Australia**

*08 / 2024*

***Group 2***

| **Name** | **Student ID** |
| --- | --- |
| ***Dylan Coon*** | ***ID - A00120427*** |
| ***Rade Cvetkoski*** | ***ID - A00097280*** |
| ***Dalton Christopher*** | ***ID - A00122255*** |
| ***Samual Allan*** | ***ID - A00108180*** |

-Rade Cvetkoski

In this section of the report I outline the background of the project as well as the post implementation review (PIR). The background part covers topics like project requirements, goals and key objectives, while the post implementation review is focused more on the achievements and lessons learned from this journey as a team and concluding if the project was a success and met its goals.

**Background**

Requirements:

GitHub - Throughout the development, the team used GitHub to share all the necessary files.

Communication and organisation - The team communicated and held meetings on Discord.

Code compiling - The prototypes were developed using C# programming language.

Middleware - RabbitMQ is set up for message brokering. It is an open-source messaging platform that facilitates connection between systems, ensuring that the users can send and receive messages.

Goals:

The main goal of this project is to build a functional application, with 3 separate forms, each being in a different domain and connected using an open source middleware.

The following domains are covered:

1. Chat application
2. Trading application
3. Contact tracing application.

***However, during the final phase of the project the team removed the trading and contact tracing app since the main focus was on further improving the UI of the chatting application. With this, the application provides plain and simple navigation for the users, avoiding any possible confusion.***

Deliverables

* Removed the single menu for all 3 applications
* Log in/sign up form
* Custom icons
* Added, Automatic Messages sent to clients when A User joins the Chat (Welcome Messages)
* Functional ListBox to list all Active Users in the chatroom
* Ability to send "Image Files|*.jpg;*.jpeg;*.png;*.bmp" -- Images have their bytes read then converted to a base64 string to handle this.
* Images in chat are scaled keeping their aspect ration and draw with padding so messages and images do not overlap
* Images sent can be clicked on by users which will open a popup displaying the image at its full scale
* Users can use the enter key to send chats if the message textbox is selected
* Logout button (safely close connection and channel)
* Use of Visual Studio Designer to design the UI visually rather than in code

**Post implementation review (PIR)**

Key objectives

| Objective | Context | Was the object met? |
| --- | --- | --- |
| Successful launch | Successfully launch the application without any bugs/downsides. | Yes |
| Project deadline | Finalise the development and reports before the project deadline | Yes |
| User interface | We aim to develop an app that is simple to use for everyone, with interactive buttons, and responds back to the using windows forms. | Yes |
| Design | We aim to find the appropriate theme and colours that suit the context and purpose of the app. | Yes |
| Using middleware as a communication tool | We aim to develop one main app with a menu that branches into the other domains using C# programming language. | Yes |
| Secure and encrypted messages/passwords | To ensure the user’s safety the app will encrypt the messages when sent and decrypt when received by the other party. | No |

Was the project a success?

***Overall the project was a success and met its goals.***

***For future reference to the director’s board, the team can work on encrypting the passwords by adopting an internationally acknowledged encryption algorithm.***

Lessons learned

* Start as soon as possible helped us achieve the goals more efficiently
* Regularly conduct meetings, communication is vital part of working as a team
* Do thorough research and learn from other organisation’s mistakes.

Challenges

* No budget
* Moving the original app from Windows PowerShell to Windows forms in the middle of development.

-Dylan Coon

In this section of the report, I will evaluate the quality, potential risks and highlight notable issues associated with the chatting application group 2 developed as part of an extension of the previous prototype assessment. The application uses Windows Forms for the user interface and RabbitMQ for real-time messages. This analysis will cover the code structure, user interface, system scalability, security vulnerabilities, message handling with RabbitMQ and user experience.

**Application Quality**

- Code structure.

· Code is organised with distinct classes handling separate functionalities, for example, ‘ChatLogin’ manages user authentication, while the ‘ChatForm’ is responsible for interface and message handling.

· Modular design makes the application clear for users.

· The code follows good practices and readability with descriptive variable names and easy to follow logic.

- User Interface

· The UI is organised to be easily understood, with text boxes for message input, buttons for sending messages and files, and text boxes for displaying chat history.

· The application allows for both text and images within the chat interface.

- RabbitMQ

· Efficient Message Handling leverages RabbitMQ for real-time message distribution.

· Queue management ensures each user is assigned a unique queue, and queues are deleted after a user has disconnected.

**Risks and issues**

| **Risk/Issue** | **Description** | **Importance and impact** | **Potential Fixes** |
| --- | --- | --- | --- |
| Error Handling | · Connection stability: No reconnection handling for possible errors with RabbitMQ connection.  · Message failures: No error handling for message loss. If a message fails to be sent the user is not notified. | High risk, high importance as no error handling can result in users continuing to use the chat interface without realising messages are not transmitted leading to frustration with data loss and potential crashes. | · Implement a reconnection strategy for RabbitMQ to handle connection drops.  · Provide user notifications if message publishing fails, ensuring that users are aware of communication issues. |
| Security Concerns | · Plaintext: Passwords are stored in plaintext without any encryption.  · Authentication: Minimal authentication methods may allow for unauthorised access. | High risk, in a real-world scenario user information is extremely important as security breaches may lead to potential lawsuits. | · Implement encryption for password storage and transmission.  · Strengthen Authentication with robust authentication mechanisms. |
| Scalability and Performance | · Message load on RabbitMQ: potential bottleneck issues in large scale usage.  · UI performance: The interface handles messages and images in a list box which may result in poor performance. | Low risk as this hasn’t been tested so we don’t know at which point the application struggles, however, in theory, as the number of users increases this could result in performance degradation on both RabbitMQ servers and Clients. | · Optimise messages: Reassess the use of exchanges and queues to manage message loads effectively.  · UI Optimization: Limit the number of messages loaded into the UI when busy to maintain performance. |
| User experience | · Overwhelming logins: Currently a notification is sent to all users when someone joins the chat which may result in the chat being drowned out by logins. | Medium as it may be inconvenient and may impact user experience as important messages may be buried beneath notifications. | · Reduce notification clutter by modifying the notification system to reduce the frequency or visibility of join messages, especially in larger groups allowing for a smoother user experience. |

* Samuel Allan

**Schedule**

This section outlines how we managed the project schedule to ensure a smooth and timely completion.

Initial Planning and Timeline Development: We started by creating a project timeline that mapped out all the key tasks and milestones. This timeline helped us see all the deliverables required for the project and efficiently manage our time. By breaking down the project into multiple phases, it ensured that each task was completed to fit our schedule.

Sprint Planning and Execution: We used an Agile approach, dividing the project into multiple two-week sprints. Each sprint focused on a specific task or feature, which allowed us to manage our work in smaller, more manageable pieces. Regular meetings helped us prioritise tasks and make sure we were on track, also allowing us to adjust if needed.

Milestone: We set key milestones for the project, such as the completion of the prototype and final testing, to track our progress. These milestones were important to keep the project on schedule, giving everyone something to work towards. When we encountered delays, we adjusted our timeline accordingly to ensure that the final delivery was still within the scheduled time.

**Costs**

This section will discuss how we can approach cost management in our project. Although the costs are theoretical due to the nature of the project, a realistic budget plan is what was aimed for.

Personnel Costs: Personnel costs can be estimated by calculating the time each team member would spend on different phases, such as design, development, and testing. Each individual task has a value assigned to it which when combined, gives an estimation on the total budget required. This method helps us to better understand how time management affects the overall costs and ensures that each phase has adequate resources assigned to it.

Software and Tool Costs: Although we did not actually purchase any software, we can consider the expenses associated with using tools like JIRA for project management, GitHub for version control, and various dev ops environments. We can also include potential future costs, such as maintenance, hosting, or updates.

Overhead: We can account for potential overhead, such as cloud services for deployment and testing. Additionally, we can include a contingency fund in our budget to cover any unexpected costs that might arise during the project. This can help us plan for potential risks and ensures that we can manage the project without financial issues.

Monitoring: Throughout the project, we can compare our estimated costs to the actual progress to make sure we were staying within our budget. This process allows us to identify any issues early and adjust as needed, helping maintain the project’s finances.