# Proof of Concept

The project team has investigated on four programming languages and whether those candidates are suitable to be chosen as a programming language to build UpStage. Java and C# have been selected as appropriate languages for UpStage development based on the research criteria to measure and judge the programming languages. However, it was not easy to choose the best only one from those two languages, because both seem incomparably suited for UpStage. Therefore the team decided to make a very simple version of UpStage. As a prototype is a demonstration of what is actually feasible with existing technology and where the technical weak spots exist, the developers are able to understand better about the development environment (Bernstein, n.d.).

As UpStage is a platform for cyberformance, it is important to consider how to implement the real time web technology. Some of team members prefer to choose C# and some prefer Java. The individual’s preference of programming language is subjective but still important. The team thought that is important for everyone to experience both languages to try to be fair. Therefore, we work in pairs and make two parts for the prototyping so that after completing the first part, the pairs can exchange the project to complete the second part. For the first part of the processes, we record the steps of implementing the function so that we can compare the simplicity of the two languages (See the appendix section). Then we can also see the other differences between those two programming languages.

## Process

* Part 1: Simulation of aspects of chat function

Two pair programmers work with the task to create a simple chat app in  
different programming language.

* + C# – Alyssa and Sia
  + Java – Jing and Joshua
* Part 2: Simulation of aspects of showing avatar

The pairs exchange the programming language to extend by adding an image  
button that adds or removes an image in real-time.

* + C# – Jing and Joshua
  + Java – Alyssa and Sia

## Criteria

1. Ease of Use  
   - How easy to code?
2. Readability  
   - How easy to understand the code written by other members? (whether they follow the coding standards)
3. Extendibility  
   - How easy to extend the code?
4. Responsiveness  
   - Always responsive?
5. Real-time  
   - How well it works when there are more than two clients at the same time?

## Findings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Prototype Criteria: | | | | |
| Language | **Ease of Use** | **Readability** | **Extendibility** | **Responsiveness** | **Real-time** |
| C# | There wasn't much use of C# in the prototype but SignalR (the API for chat) was fairly easy to use. The pair programmers involved with chat were familiar with SignalR so it was easily implemented. The second pair completed the extension in a short period mainly because JavaScript was the main language involved so this only shows that C# is not any better for coding only that C# easily supports implementation of this prototype’s functions. | The steps to complete the prototypes were documented and made it easier to understand what had been done. However when extending the code it was difficult to understand coding. In C# the unfamiliar layout made it hard to read. | Adding the avatar button was easily done | Was always responsive | yes |
| Java | Most of part 1 consisted of researching for a suitable API for the chat function (Atmosphere was used). Chat was easily developed but after atmosphere was used it became difficult to use the code. This is because Atmosphere added a large amount of code within the projects classes. Using its commands were simple but the structure of the classes changed a great deal confusing the students about where to make changes to add the extra feature (adding avatar button). This process took the team hours and a lot of effort and in the end failed to produce the functionality required. | In Java it was the Atmosphere's code that was difficult to read. | The code was difficult to extend due to the complex coding in the Atmosphere API (used for chat). | Was always responsive | The chat prototype was real-time but extension (i.e. adding avatar button) was not real-time. In the end it was not solved. |