# Development testing

This chapter will focus on the implementation of the test application described in chapter (TODO: crossref testapp methodology). Each framework has its own subsection where I describe every aspect of the development and thoughts I made during the process. Each section has its own summary, and finally, there is a conclusion giving a nuanced look at what framework solves the different tasks of a real-time application best.

## Socket.IO

Socket.io is a module for Node.js (TODO: nodejs) that provides real-time through pure JavaScript on both server and client. It has been around since 2011 (TODO: last commitpage)[[1]](#footnote-1), and it aims to provide clean and simple real-time across all platforms:

*“Socket.IOaims to make realtime apps possible in every browser and mobile device, blurring the difference between the different transport mechanisms”.*

While it hasn’t reached 1.0 yet (TODO: check), it is used in production code by several companies, and it is considered stable. Perhaps one of the most “famous” applications that use Socket.IO is Trello – the online “Scrum Board”[[2]](#footnote-2).

### Why I chose it

Node.js is increasingly popular, and the idea of using JavaScript on the server is very exciting! Over the past couple of years, there has been a dramatic change in the way developers think of JavaScript (TODO: sources). Therefore, it was only natural that I chose at least one framework that uses Node.js as server.

Though there are several modules for Node that provides real-time (TODO: link to node modules), Socket.IO stands out from the crowd. It seems to have the largest community, as it is frequently featured at conferences and generally mentioned many times in traditional forums like Stackoverflow[[3]](#footnote-3).

Furthermore, Socket.IO feels like more than just a Node module. It has its own homepage (TODO: homepage) with some examples and demos–all presented in a good looking and easy to understand fashion. I feel this gives Socket.IO a more professional impression, which makes it stand out even more from some of the other modules that exist that seem more like something someone threw together in a hurry.

Socket.IO doesn’t have a lot of documentation, but what it has gives users a quick overview of the module and how to use it. The API documentation (TODO: docs) uses code samples, which I find more useful than a so-called “wall of text”. There is also a wiki page (TODO: wiki) to give information beyond the API documentation.

As stated in the quote from Socket.IO’s homepage in the introduction to this section, Socket.IO strives to blur the difference between the different transport mechanisms. WebSockets is the preferred transport, but if the client doesn’t support it, Socket.IO will fall back gracefully[[4]](#footnote-4) to one of the following transports:

* Adobe Flash Socket (TODO: source), which uses, surprise, Flash to establish a TCP socket connection between the client and the server, thus “mimicking” a WebSocket connection.
* Ajax multipart streaming (TODO: source): An alternative streaming technique to the forever frame technique described in section (TODO: crossref AND should I write this in the essaypart?).
* Forever Frame
* JSONP Polling, which is polling with data type set to JSONP. This allows cross domain requests; something that is not allowed in normal HTTP Polling (TODO: same source as multipart).

## Lightstreamer

## Play Framework

## SignalR

## Meteor

## Conclusion

1. 0.7 preview was released May 5th 2011 [↑](#footnote-ref-1)
2. <https://trello.com/> [↑](#footnote-ref-2)
3. [www.stackoverflow.com](http://www.stackoverflow.com) [↑](#footnote-ref-3)
4. The fallback happens ”behind the scenes” so that developers do not need to worry about it. [↑](#footnote-ref-4)