Literary Worlds Javascript Client

John Lewis, Owen Watson, Tim Cunningham

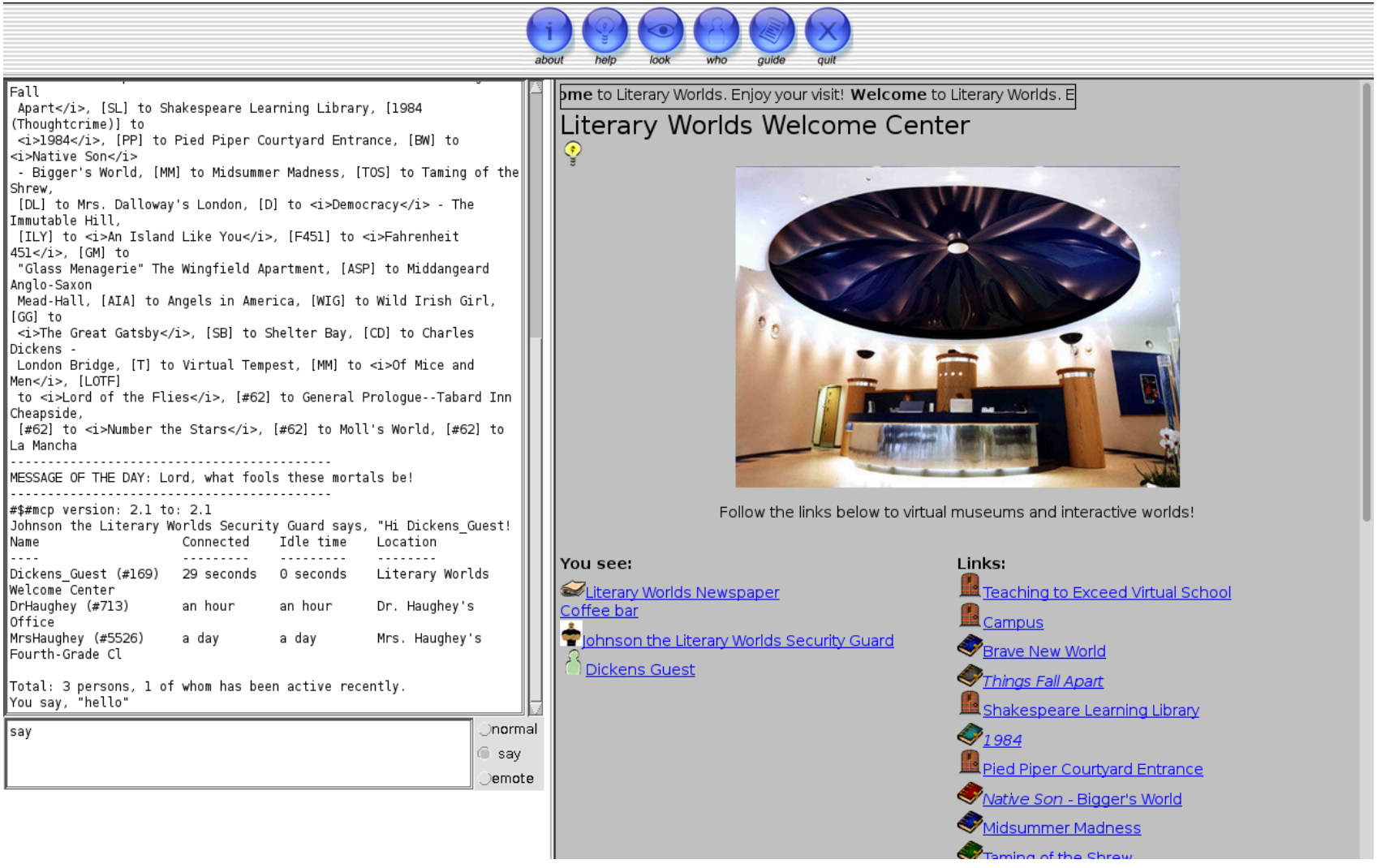
November 10, 2014

Table of contents

Background

- Literary Worlds is an multiuser text based game used for English education at WMU.
- These games are known as MOOs (Multiuser Object Oriented), or MUDs (Multi User Domains)
- Most of these games provide a telnet interface for playing, Literary Worlds uses enCore with MOOtcan, a Java applet.
- The aim of this project is to provide a drop-in replacement user interface in Javascript.

enCore Xpress interface



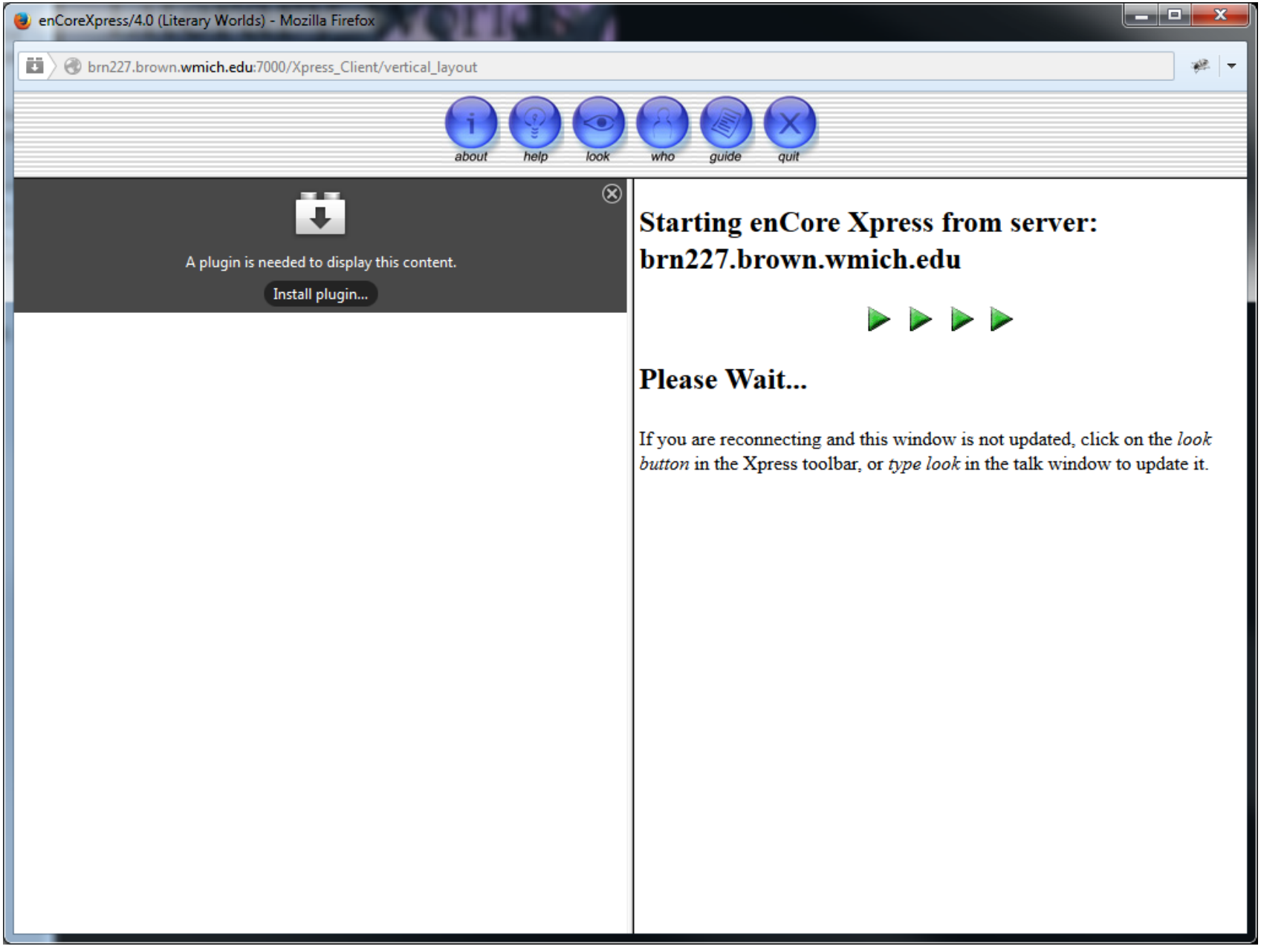
Client

- Client: Allen Webb
- Professor of Comparative Literature and Postcolonial Studies at Western Michigan University's Department of English
- In 2003, Robert Rozema, a PhD student under Allen Webb created the prototype Literary World.
- Several virtual worlds were created that form the Literary Worlds project.
- The Literary Worlds project was one of seven projects to receive funding from the Western Michigan University President's Innovation Fund.

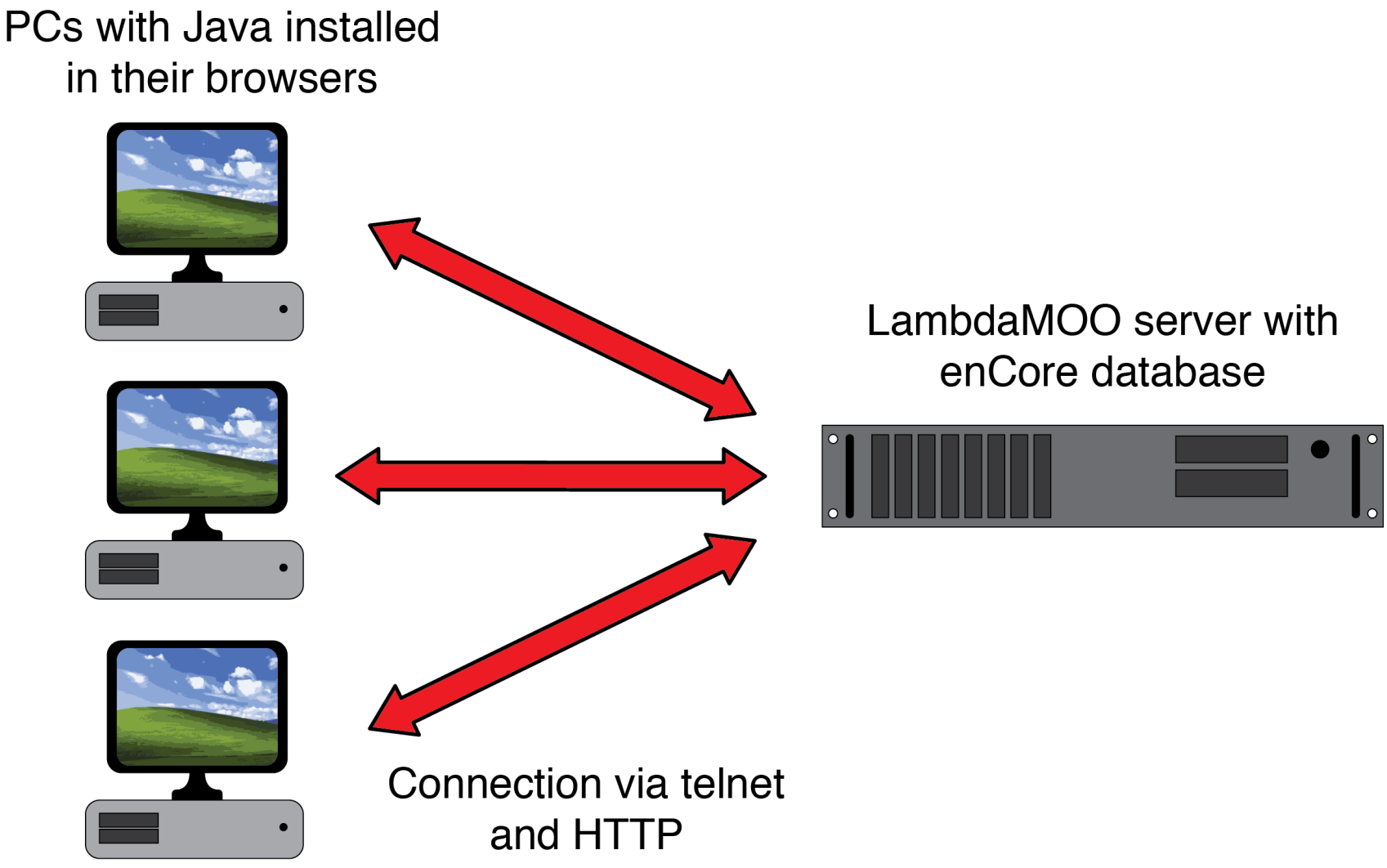
Technical Introduction

- LambdaMOO is a MUD server software package(as well as a particular MUD), Literary Worlds uses the server software.
- enCore Xpress is an graphical interface and MUD database package that works with the LambdaMOO server to provide a browser based text client using a Java applet as well as a graphical, mouse driven interface.
- Literary Worlds uses version 4 of enCore, there is a version 5 of enCore that attempts to provide a web interface without a Java applet, however it is beta software and hasn't been actively developed since 2006.

Plugin Missing



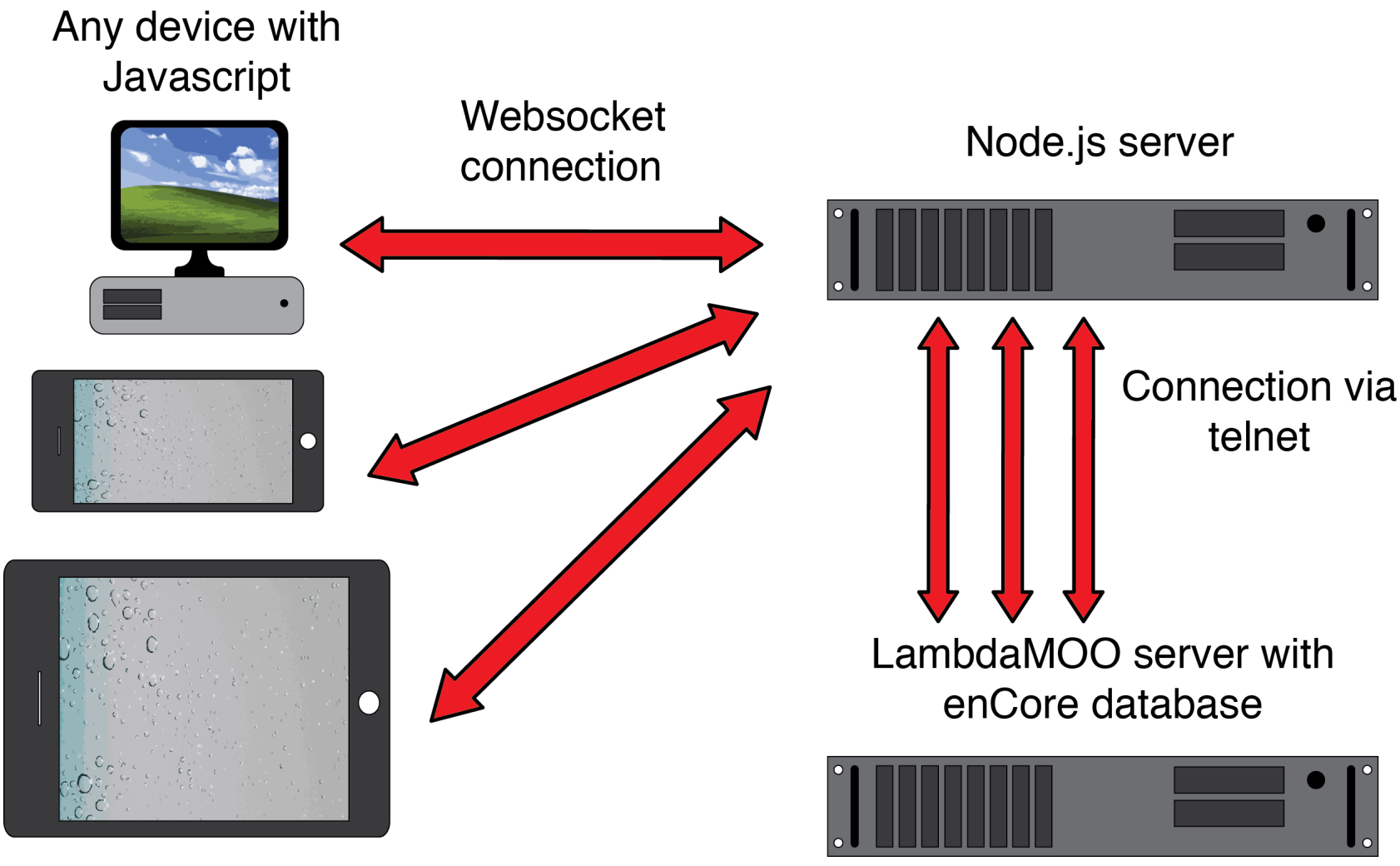
Client-Server Diagram



Client-Server Design

- It is not possible to initiate a raw TCP connection in clientside Javascript, unlike Java applets.
- Therefore, we need an intermediary server to connect over TCP, as well as to the browser client, and send data back and forth between the two.
- This server must also support multiple concurrent users and handle asynchronous tasks and events.

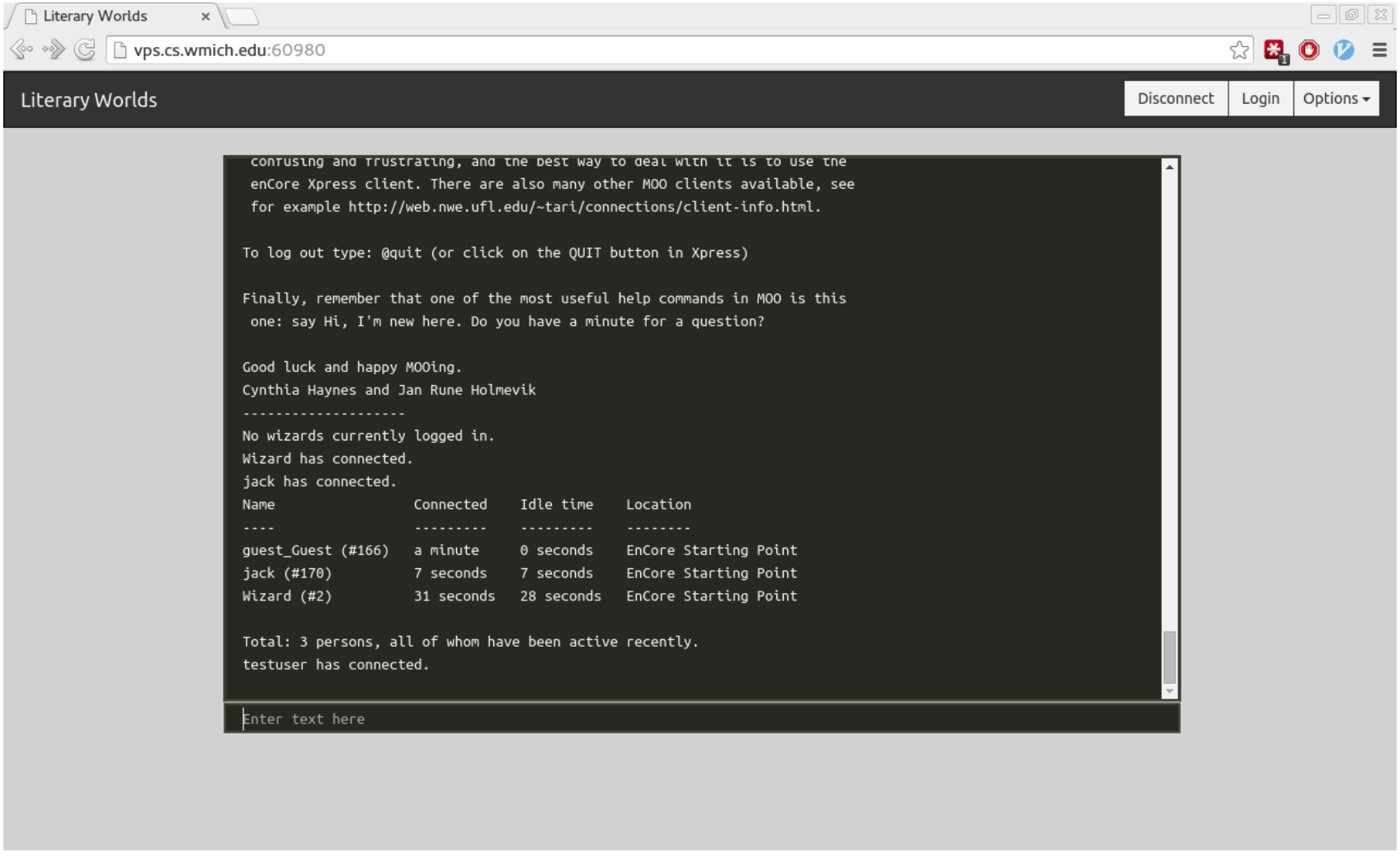
Client-Server Diagram



Client Technology

- Socket.io
  - Allows realtime communication between a server program and browser clients.
- Backbone
  - Backbone is a templating library for Javascript web applications
- Bootstrap
  - Bootstrap is a frontend layout toolkit
- Coffeescript
  - Coffeescript is a simple language that translates to Javascript

Basic Text Interface



Server Technology

- Node.js
  - Node.js is a server side runtime that uses the asynchronous features of Javascript to build web application servers. It provides a rich set of networking libraries, and uses the Google Chrome Javascript engine, V8.
  - It is the first class citizen for socket.io, so it was natural to choose it for the server.
- Express
  - Express is a web app framework for Node.js
  - In provides tools such as easy interfaces for REST, URL routing, cookies, etc. similar to what Ruby on Rails does for the Ruby language.
- Socket.io

Stories

- Text Mode
  - Client side Javascript MUD client
  - This is reason an intermediary server is needed, to handle the telnet connection on behalf of each client.
- Graphical Mode
  - This work is already done in enCore Xpress, there is no need to reinvent it.
  - The new web application will simply need to serve the existing enCore Xpress interface in a coherent way.

Continuous Integration

- Purpose
  - Continuous integration aims to merge code from a developer's working copy many times a day into a shared mainline.
  - Before any code is merged a, build server runs unit tests on the code, and if passed will be merged into the mainline.f
  - This helps to improve the quality of the software and reduce the time to deliver it.
  - The end goal of this process is to be able to automatically build and deploy the software whenever all of the tests are passing
- Possible Tools
  - Jenkins
  - CICircle
  - TeamCity
  - Hudson

Jenkins

- Pros:
  - Widely used and documented
  - open sourced
  - Wide range of support for different systems
  - Can be extended with plugins
  - Github integration
- Cons:
  - Manual setup required
  - Have to get a server to run Jenkins

Unit Testing

- Framework
- Matching Tools

Security

- Telnet is plain text in transport, so it is inherently vulnerable to eavesdropping.
- For the initial work on the project, security is not a main concern.
- In the future, the telnet server can be protected from external network connections using containerization such as docker
- This way only the Node.js server can connect via telnet, and user data can be protected with SSL in transport.
- This strategy would disable the legacy enCore Xpress interface