“Mobilise” Redevelopment High Level Specification

Functional descriptions and estimates

Draft: 10 October 2020

Created by Paul Rohrlach, Gloria W. and Helen Varley Jamieson

Table of Contents

Revision History

Gloria	2020-08-30	Initial version
Helen	2020-08-31	
Paul	2020-09-01	Formatting and grammatical changes
Helen	2020-09-02	small comments
Helen	2020-09-03	added some more comments & questions, & expanded on the opening section about what UpStage is
Paul & Gloria		further refinement of details
Helen	2020-10-10	formatting and PDF

1. What is UpStage?

UpStage is an online collaborative performance and communication platform used by artists and students for close to seventeen years. It allows for the real-time manipulation of digital media in a browser-based interactive stage environment, using digital media including images, animation, audio, text-to-speech, text, live audio-visual streams, and live drawing. Artists and audience present on the virtual stage see and hear everything at the same time, and can interact in real time via a text chat.

This platform has been used as an artistic tool, a teaching environment, and a creative and entertainment platform to encourage discussion and creative exploration of everything from political and social issues to fun topics. Accessibility for audiences and artists is a key feature - the platform is browser-based and requires no additional download or installation to create or attend performances. It’s a profound, complex platform that is needed now more than ever, in this time in our history when so many people cannot physically interact, or are struggling to adapt to online interaction.

A platform rebuild is long overdue. The purpose of the rebuild is to bring UpStage up to date with current technology, most importantly removing the dependency on Flash & making the software compatible with mobile devices. An installation package will be created for those who wish to set up their own UpStage instance on their own server, along with appropriate technical documentation to support both installation and ongoing maintenance.

2. Components

Upstage has these high-level components, assets, and attributes:

Static content: considered by our server-side code to be assets that are executed or displayed by the web browser.

- image files: still and animated graphics, for example animated image files that display different positional “frames”, appearing as motion once loaded into a browser. These are used on the stage as avatars, backdrops, and props.
- audio files: music and sound effects in common formats.

Dynamic content:

- text chat: text entered in real-time via the chat window, by players and audience, and can be read by a text-to-speech program in the browser.
- “voice” files that add text-to-speech support for new voice types and pitches, accents, etc.
- live drawing: real-time creation of drawings by individual artists or collaboratively, directly onto the stage.
- live audio-visual streams: webcam feeds streamed in real-time to the stage.
- external media: ephemeral content called in for a specific period of time (video, image, audio) as an embedded function

Server-side “event processors” able to transfer avatar movement data, speech data, prop data, other related data to each connected device, and able to receive and send these events from/to every connected device.

Server-side static content servers, which serve this content to all connected devices. This will not only use SSL, but will also show creator attribution and allow the creator to control access to their avatars, voices, and other media used in Upstage.

Browser-side avatar creation tools, which allow a performer to create and edit avatars within reason. The avatar is first created with an Open Source avatar creation tool. The standard file formats generated in Open Source will be supported in UpStage.

The “live stage”, which is populated with assets prior to the performance, and is the main interactive “console” for Upstage avatars. This development is most likely all in some form of Javascript, HTML5, and possibly WebGL, based on performance and difficulty of support.

The live chat, where attendees can interact, and where performers can see and react to chat, as well as participate in conversations.

The “workshop” or backstage area - where media can be uploaded, assigned to stages, some editing e.g. selecting voices; stages are created & edited; player profiles managed (at the moment just people’s passwords, not any kind of visible profile) and other administration tasks.

3. Network and I/O Layers

The performance of the Upstage platform is a vital part of its success. To make an Upstage live performance ideal irrespective of location or type of computer, we are considering the following:

The serving of live events, at its lowest level, will always need to open a socket, transfer data, and close a socket. This is how all protocols work on all computers at their lowest levels. The act of opening and closing this connection is an expensive (CPU-intensive) operation for each sending and receiving computer to do repeatedly. This is why we are considering methods and network protocols which allow for “long-lived” “self-healing” connections between “brokers” (site-to-site servers which “sync” with each other), and between servers and performers/participants.

To accomplish this we will consider several solutions: MQTT using Mosquitto, Websocket using POSIX threads/lightweight processes without global locking mechanisms, and other socket I/O tools/libraries. Experimenting with this, choosing the right solution, and then implementation will take approximately two months for one developer at 40 hours/week.

The static content server can be considered in the I/O layer as well, for the purposes of a high-level overview. It is much easier to implement and is a well-understood problem with common solutions. We will be using the Nginx/UWSGI/Flask stack, with a SqlAlchemy/Postgresql or CouchDB persistent data layer, to possibly handle user-based permissions. There will be consideration and experimentation with newer token-based permissions models, and to allow for time to work with these, we will need approximately one month for one developer at 40 hours/week.

4. Communication/message layer between Performers and Participants

The Upstage “protocol” layer has to be examined and refactored/reimplemented to reflect new features and performance needs. Defining the messages which travel from the server (broker) to the server won’t be necessary if MQTT is chosen. For other solutions, defining the messages servers need to “sync” with each other will be necessary. Defining messages to update dialogue, avatar movement and interaction, speech, and interaction with scenery and other assets will be quite a bit of work. After implementation, it will be regularly updated and “tweaked” for optimal performance and new messaging. This will take approximately two months of implementation and one month of experimentation for one developer at 40 hours/week.

5. Browser support of the Stage, Character Speech, Movement and Interaction

This is an intensive part of this project, using combinations of two or three languages, several tools and frameworks, and possibly OpenGL tools supported by common browsers.
Implementing a fully functional “stage”, which is really an open canvas of predefined objects interacting in a variety of ways based on predefined rules, involves implementation, writing unit tests, and creating “baseline” characters, dialogues, and interactions. This will require a small team of browser/front end developers (3 to 5) four to five months at 40 hours/week.

6. Packaging for Ease of Use

Developers will have to work together to “package” this in a way that is as easy to set up, configure and install/run. “Out-of-the-box” basic execution and ease of configuring multi-broker configuration, with basic character configuration and functionality is our goal. This will take the entire team’s effort for approximately three weeks at 40 hours a week.

7. Support and Maintenance

To properly support and maintain this platform after release, developers will need to perform bug fixes and add reasonably scoped new features. This would require at least one front end developer and at least one back end developer part-time (20 hours/week) for several months (up to 18 months or more) after release.

The project will produce technical documentation for developers joining the project and installation manual for anyone wishing to install their own UpStage server, as well as a user manual aimed at a non-technical audience. The team will also work on a business model, planning to use Upstage to bring in revenue streams to sustain the project past this period.