Functional Specifications of Acted Features

Development Milestone: **M6**
Main name: **Chat-room**
Date: 27/05/2016

Current version: **V1.1**
Current status: DRAFT (will move to final when new screenshots ready)

Approved by:
Approval date:

Reviewed by:

History

Version	Date	By	Comment
V1.0	17/05/2016	Laurent	DRAFT for quotation
V1.1	27/05/2016	Damien	DRAFT Rewriting

Table of content

1. Chat-room

The main purpose of this feature is to allow Acted users to chat with each other. This allows a 1-to-1 conversation to be initiated or continued between an ARTIST and a CLIENT.
Booth parties needs to be authenticated, in order to be allowed to enter into discussion

In the following part, Conversation, Thread, Chat-Room or similar terms all mean's the same thing. This may be confusing sometimes, and will need to be cleaned-up at later stage.

When a Quotation request has been send by a Client to a specific artist, a new Conversation thread is created for the related EVENT.
The new event ID is created, and shall start with a minimum value of 1200 when the database is created. Afterwards it will be automatically incremented.
A copy of the ID will be created in an additional column as a String that will be used for display. That String will initially contain the ID number, but will be changed in the future with a unique set of 6-8 character, like the booking confirmation numbers you may get from flight or train bookings. That column will be indexed as it will be used in various searches in the future.

When a user clicks on the "Messages" button, from the **dashboard page**, a page should appear, allowing the user to read his previous messages and to type some new messages.

Screen shots for the Chatroom page for reference only

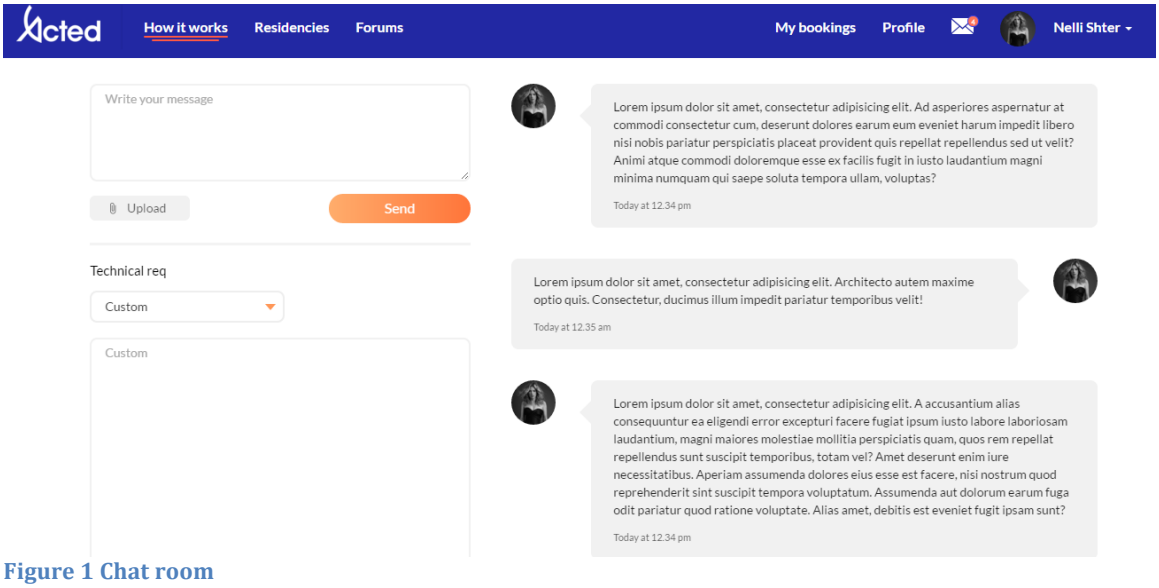

Figure 1 Chat room

Input

- The design mockup of the quote request popup form, in the "Chat-room.html" file, from the Mockup repository.
- Acted database.

Expected output

A popup, respecting the page design of the "Chat-room.html" file, behaving as requested in the core development part bellow.

Core development

A user will be able to type a message and upload a file. Then by clicking on the "Send" button, the message will appear on the right side of the window. Each message will appear with the date and hour it has been saved in the system.

- **Message:** max 512 characters text
- **Upload:** the uploaded file size should be limited to a configurable value.

2. Update of the design files

New design files will be delivered during the development of that milestone, and need to be integrated.
This includes some resizing, and the final Acted logo file.
Retina screen high resolution images and browser detection has also been added

We need a fully documented process for integrating any further design files modifications.
In particular, if you need to do some changes on your side in any of the CSS, JS files, we need to do it in coordination with the Design team, so that we create a single master for both teams.

3. Others

3.1. Bundles names

When applicable, a specific bundle name must be defined, unless it is too much interacting with the rest of the application.

3.2. Constraints

No commercial or licensed frameworks or API should be used to achieve this bundle. If any is required, this must first be approved by one of the directors Acted Labs Limited

Only open-source codes or frameworks could be integrated in the final developments.
For Open source components, the terms and conditions of the license must be listed in the final documentation in order to ensure GPL compliancy. The list must include the previously designed and already implemented components from Millstones M1 to M4

3.3. Code commenting and versioning

In each source code file (PHP mainly but no limited to these files) a proper header file must be added. The header is made of two distinc parts:

Part 1: In the header it shall mention the Acted Labs Limited copyright
Part 2: Version number in the form of Vx.y.z

Where
X is major (new functionality, incompatibility with previous major version)
Y is minor (new functionalities or enhancement but backward compatibility ensured)
Z is patch (bug fixes)

Part 1 will be the same for all files, and we will create an MD5 signature of that section for our own use. We will provide the master text during the development.

Each major part of the code including every function must be documented. It needs to explain the functionality and any necessary details for comprehension.
Things that are self-explanatory do not need to be documented but this is an exception rather than the rule

3.4. Database changes or additions

Every required change into the database structure, index, data sizing must be discussed with Acted, and approved by Acted. The same is true for any update or upgrade of the DB version or any related component, library, connector, etc…

The main reason is that there are future expected functionalities that will be developed, and we want to start understanding the total sizing of the solution, replication requirements, possible optimisations such as using SSD disks, etc…

We are also happy to discuss any possible improvements when possible scaling issues are foreseen during the development.

3.5. Database fixtures for testing purposes

Sufficient amount of data must be available into the database (via fixtures) in order to ensure efficient testing of that feature.
The additional data entered during testing shall be kept for future use when appropriate, so that it can be re-played after a database DROP.

3.6. Data migration from now on

From this milestone, we will keep all the data entered during testing, and therefore data migration must be planned accordingly.

3.7. Test materials

In order to test the development of the 2 scenario (authenticated vs. unauthenticated users), you could implement a very simple page with 2 separate forms if there is no other way to achieve proper testing:

- First form will send a quote request page as an unauthenticated user (no information in the Session), with an Artist ID as a parameter
- Second form will send a quote request as an authenticated user, supplying an user Id (saved in the Session) and an Artist ID parameter.

3.8. Documentation

A properly formatted Word document will be required to describe all the useful info about the features of this specification, to explain, if it applies:

- What is the main goal of the feature?
- What are the feature inputs?
- What are the feature outputs?
- What are the main business rules implemented?

- Define a detailed “How it works” schema.
- Documents the features that are not implemented here, but part of a “next” milestone
- Document for further reference the “prepared for future use” features and details. In particular (but not limited to) the LOCALE related ones.
- Which bundle handles each of the feature(s)?
- What are the main classes?
- What are the existing services used, and services created in that feature?
- What are the Twig templates?
- What are the add-on bundles you installed, including licensing information that applies?
- What are the commands to run to install the feature?
- What are the commands to run to test the feature?
- How to test the feature?
- What are the deployment server requirements? In particular any specific (minimum version) used during development and validation
- GIT related information’s. In particular, for testing purposes and long term tracking, please indicate key GIT tags (and/or Branches) that relates to fully functional deliverables.
- Database requirement explained. In particular Indexes, foreseen performance issues or challenges