# Milestone 3 – Final project definition

###### Volunteer Movie Scheduler

### System Description

##### Our company, Achievatron Unlimited, will create a piece of software designed to assist the Producer of a small-scale film production company. These companies rely heavily on volunteer cast or crew members as well as volunteered equipment in order to film the scenes.

##### Producers of such companies are expected to efficiently schedule filming around the time constraints of the available volunteers and equipment in order to produce the film in the shortest amount of time possible. Our software aims to make the producer’s job easier by keeping records of all cast/crew members and equipment as well as the times they are available, the scenes which are required for the movie (both completed and yet to be shot), and what each scene requires. This will save the producer time by allowing him or her to easily create a schedule, or even automatically generate an appropriate schedule given all of the time constraints.

### Buisness Case

##### A significant amount of the responsibility for the organisation of a movie project falls to the producer, once a shooting begins. It is up to them to decide when each scene should be shot, keep track of progress on the film, inform the required cast and crew of when they will be required, ensure that all the necessary equipment (lighting, cameras, microphones, generators) is present for any given scene, and keep everything flowing on any given day of filming. This workload is increased significantly when producing a movie which is reliant of volunteers or volunteered equipment and has, to some extent, to work itself around their schedules.

##### In order to ease the complexity of this task of organization our firm will design and build a piece of software which will assist in the collection of all the information needed to build a shooting itinerary for an entire movie project, including the scenes to be shot, the volunteers who will form the cast and crew, the equipment needed to be shot and the volunteer/equipment requirements of the scenes. With this information the producer will have a visual and automatic means of bringing together this information into a workable itinerary. The system will also facilitate communication between the producer and the volunteers by allowing the emailing of all relevant parties required for given scenes (in the case, for instance, of an unworkable itinerary), and will feature a web applet that will allow the volunteers to update their information from across the web, should their availability or contact info change.

### User-Level Goals

The producer and volunteers would have access to the system; however, both have different requirements. This can be achieved by simply limiting the privileges of the volunteer. The producer’s expectations are as follows:

He or she needs to be able to

* add, remove, and edit new cast/ crew members into the database
* add, remove and edit new equipment to the database
* freely edit the script and scene information as well as their requirements
* track progress easily
* customize his or her own schedule, with our system’s assistance
* generate a schedule based on the availabilities of persons and equipment
* give each volunteer a personal schedule detailing which scenes he or she will be working on

The producer also expects to be informed of any conflicts, and be given possible solutions to deal with issues in a timely manner.

The volunteer expects to:

* have access to their schedule
* be able to edit their available time
* be able to edit their contact information

### user scenarios

##### In our project we have identified one major use case and one minor use case. The primary users will be one or more producers whose primary uses of the software will be to organize scheduling and resolving conflicts involving volunteers/equipment. The secondary use of the software will be for volunteers to easily edit contact info and availability to resolve conflicts.

##### **Use Case**: Enter new volunteer/equipment information

##### Actors: Producers

##### Goal: To enter new information into the database regarding volunteer availabilities

##### Activity: A producer receives a form with the volunteer's skills as they relate to the film, their availability and contact information. They then are able to quickly enter it in large quantities.

##### Quality: Must be efficient, and accurate

##### **Use Case**: Enter new equipment information

##### Actors: Producers

##### Goal: TO enter new equipment information into the database

##### Activity: The producer receives equipment rental information time availability, and they enter the attributes and times for the equipment. Should be efficient

##### Quality: Efficient and accurate

##### **Use Case**: Enter/Edit scene requirements

##### Actors: Producers

##### Goal: Create and edit what each scene needs.

##### Activity: Allow the producers to check off what type of people and equipment is needed for a particular scene, so that a schedule can be generated. It should also allow the editing of requirements.

##### Quality: Efficient and simple

##### **Use Case**: Edit/Update Information

##### Actors: Producers/Volunteers

##### Goal: Edit information of the volunteer/equipment information

##### Activity: Allow the editing of volunteer/equipment availability by the producer or the Volunteers through the web service

##### Quality: Reliable, simple, and consistent

##### **Use Case**: View Schedule

##### Actors: Producers

##### Goal: Examine the current schedule

##### Activity: Once the volunteer availability and scene requirements are input, allow the producer to have an 'at a glance' view of the schedule.

##### Quality: Must 100% correct, concise as possible and reliable.

##### **Use Case**: Export schedule

##### Actors: Producers

##### Goal: Get a file containing the schedule

##### Activity: Allow the producer to export the schedule data into a xls, or similar format that would allow printing the schedule.

##### Quality: Must be easily human readable.

##### **Use Case:** Conflict resolution

##### Actors: Producers

##### Goal: Resolve conflicts in the schedule

##### Activity: Allow the producers to view conflicts, and make decisions, such as to send everyone involved emails, or to generate a list of their phone numbers.

##### Quality: Reliable and simple (For the user)

### Scope document

##### Completed system will:

* Allow volunteers to create and update a profile detailing their information and availability as well as their status of either cast or crew.
* Allow the producer to label cast and crew members with levels of priority in order to better manage scheduling conflicts.
* Allow the producer to create and update an equipment information and availability list.
* Allow the producer to schedule volunteers for scenes.
* Notify the producer in the event of a scheduling conflict.
* Notify the volunteers in the event of a scheduling conflict.
* Attempt to manage and resolve the conflicts, letting the producer know his best options before he can make a judgment call about who will be in a scene.
* Create a schedule that both volunteers and the producer can view by logging into their profile.

##### Completed system will not:

* Allow volunteers to schedule anything in any way.
* Allow volunteers to access the equipment list in any way.
* Magically solve conflicts; it will give a list of best options to be used at the discretion of the producer.

### Project plan – rough estimates

|  |  |
| --- | --- |
| Task | Estimated Time |
| Create and populate Database/API | 4h |
| Code: Business objects | 3-4h |
| Code: Volunteer/Equipment Availability | 8h |
| Code: Scene Requirements | 8h |
| Code: Schedule generation | 10h |
| Code: Conflict Resolution | 9h |
| Code: Applet Prototype | 10h |
| Code: Central UI | 3h |
| Usability testing | 2h |
| Design and Final report | 5h |

### user involvement plan

##### Dead Prairies Productions is a movie company located in Swift Current, SK who one of our group members worked with a few summers ago. A volunteer producer and various cast/crew members from this production group have agreed to help test our Movie Scheduler product. This will occur over three phases during the development of the project.

##### **PHASE 1: PROTOTYPE**

##### This phase involves testing the usability and flow of a basic GUI with the prototype movie scheduler.

##### The producer and a few cast or crew members will test out the prototype. The main benefit of this phase is that the volunteers will be able to offer suggestions towards the look and feel of the interface which will aid in improving overall user experience as we continue to develop.

##### Users: 1 Producer, 2-3 cast / crew members

##### Time required: 15-30 minutes.

##### Date: Early October upon completion of a prototype with basic functionality.

### **Phase 2: Implementation**

##### This phase involves users testing some major features of the project during the implementation. The Producer and various cast/crew will be asked to use the movie scheduler and cast/crew availability interfaces. This will allow us to catch any major issues that we may have overlooked.

##### **Users:** 1 Producer, 5-6 cast / crew members

##### **Time required:** 30-60 minutes for the producer, 15-30 minutes for each cast/crew member

##### **Date:** Early - Mid November, upon completion of major features, before actual completion and deployment of our product.

##### **PHASE 3: FINISHED PRODUCT**

##### This phase involves a trial run - Dead Prairie Productions has agreed to use our Movie Scheduler for a short film. This will allow us to evaluate the product before the development process is over, as we will work with users to fine-tune the product - with the user input we can make slight tweaks to the interface to improve the overall flow and feel of the program.

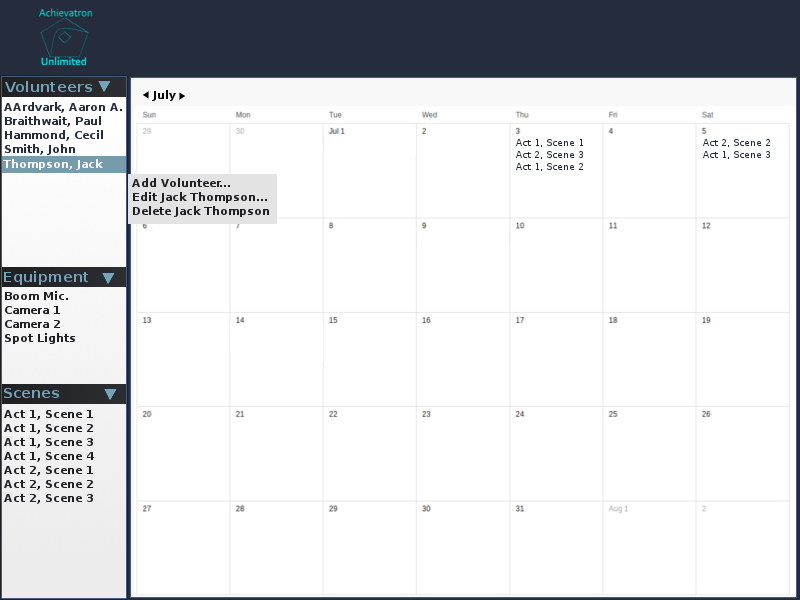
##### Users: 1 Producer, a number of cast / crew.

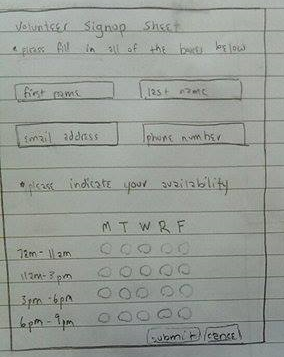
##### Time required: The length of the filming of a short film (3-4 weeks).

##### Date: Mid November - Early December, before deployment of our product.

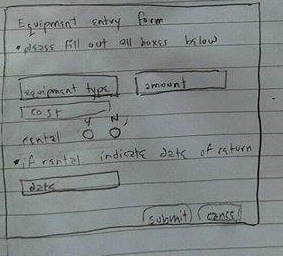
### low fidelity prototypes

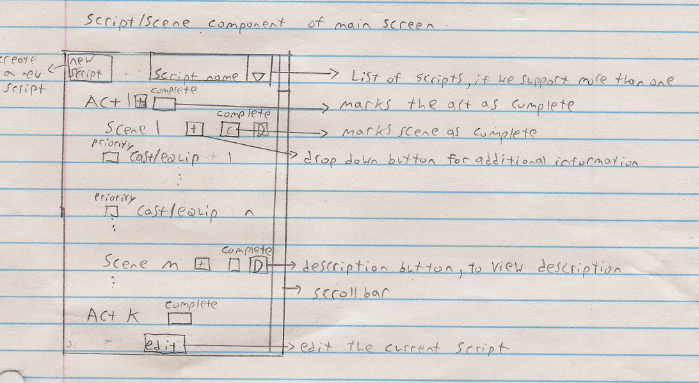
The main screen:

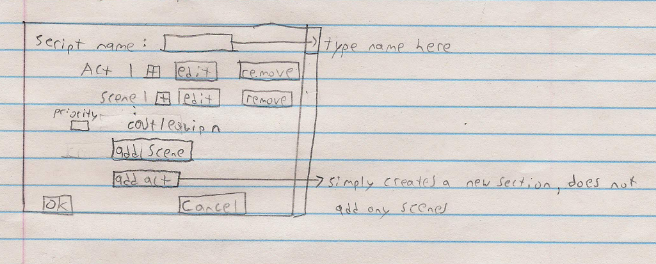




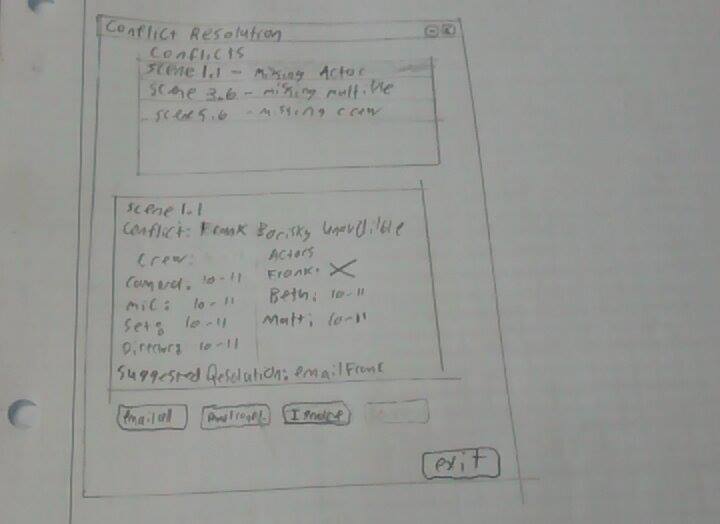
Volunteer/Equipment info

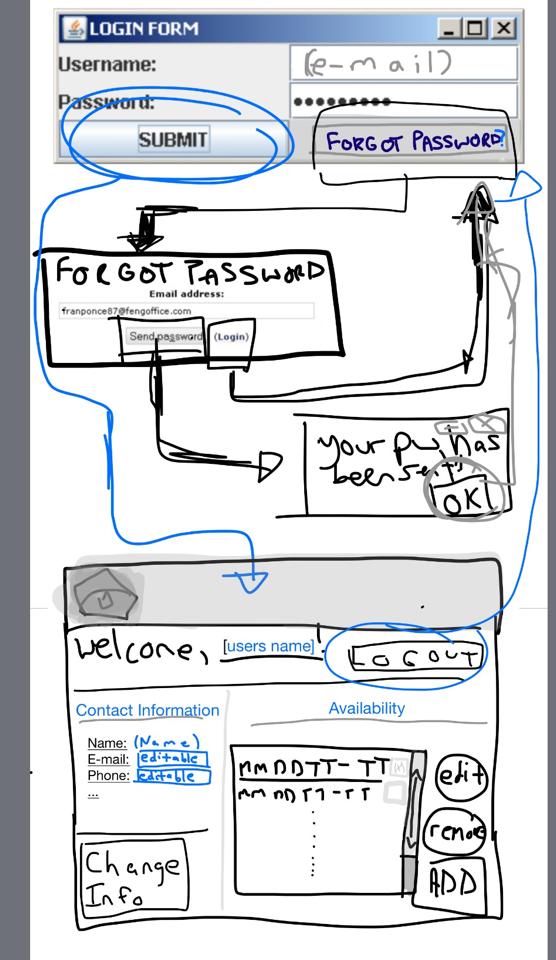


Scene requirements



Conflict Resolution



Web Applet Prototype

### project management report

Time spent:

|  |  |  |
| --- | --- | --- |
| Iain | 10h | Database API and Github |
| Mitchell | 7h | Administrative/project planning and putting milestone together |
| John | 3h | UML diagrams/Scope report |
| Ryan | 6h | Logo, milestone tasks |
| Matt | 6h | User involvement, prototype design |

##### A potential risk of our project is that we will not be able to implement all features, but the division of work requires that all components be at least functional for the proper implementation of the successful use cases. Administrative time is overseen by Mitchell, and he’s aggressive about time management, so worrying about administrative time is not an issue.

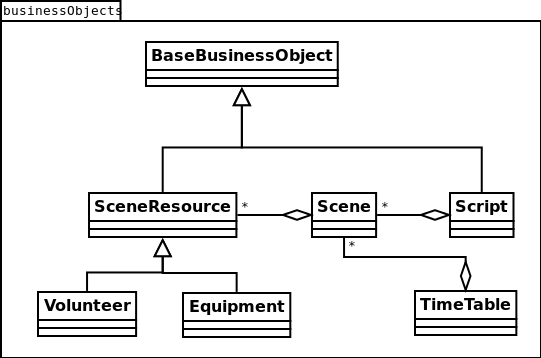
### project plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task Description | Who? | Time Expected | Actual Time | Expected Completion | Status |
| Set up Github | Iain | 1.5h | 2h | Sept 22nd | Complete |
| Divide up work appropriately | All | .5h | .5h | Sept 22nd | Complete |
| Design the database API | Iain | 3h | 6h | Oct 1sts | In Progress |
| Create example databases | Iain | 1h |  | Oct 1st | In progress |
| Create Buisness Objects |  | 3.5h |  | Oct 4th | In progress |
| UML Modelling of system | All | 2h |  | Oct 15th | Not Started |
| Decide on Conventions | All | 1h | 1h | Oct 1st | Completed |
| Code: Volunteer/equipment Availability | John | 8h |  | Oct 12th (Prototype) | Not Started |
| Code: Scene Requirements | Ryan | 8h |  | Oct 12th (Prototype) | Not Started |
| Code: Schedule Generation | Iain | 10h |  | “ | Not Started |
| Code: Conflict Resolution | Mitchell | 9h |  | “ | Not Started |
| Code: Applet Prototype | Matt | 7h |  | “ | Not Started |
| Usability Testing | ALL |  |  |  | Not Started |
| Code: Central UI |  | 2h |  | Oct 12th | Not Started |
| 4. Use cases | Ryan | 3h |  | Oct 12th | Not Started |
| 4. Specifications |  | 2h |  | “ | Not Started |
| 4. Use Case Diagrams | Mitchell | 1h |  | “ | Not Started |
| 4. Domain Model | Iain | 1.5h |  | “ | Not Started |
| 4. Glossary | John | 1h |  | “ | Not Started |
| 4. System Sequence Diagram | All |  |  | “ | Not Started |
| 4. Contracts | Matt | 2h |  | “ | Not Started |
| 4. Implementation | All | (Derived from above) |  | “ | Not Started |
| 4. Project Plan | Mitchell | 1h |  | “ | Not Started |
| 4. Meeting Minutes | Mitchell | .25h |  | “ | Not Started |
| 4. Ensure proper format for hand in | Mitchell | 1h |  | “ | Not Started |

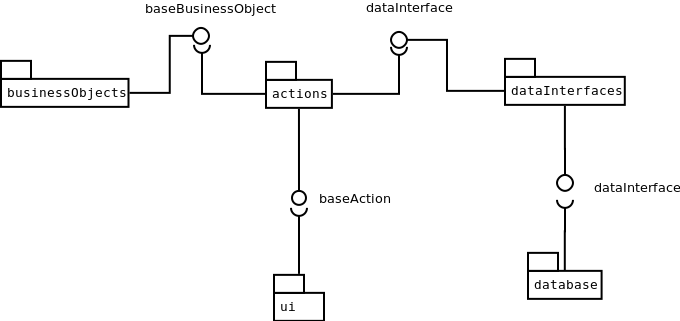
### toy example

##### For our Toy Example, we implemented a main window, and created the database, as well as establishing the database connection. We have also done some UML modelling of the implementation as well as the database:

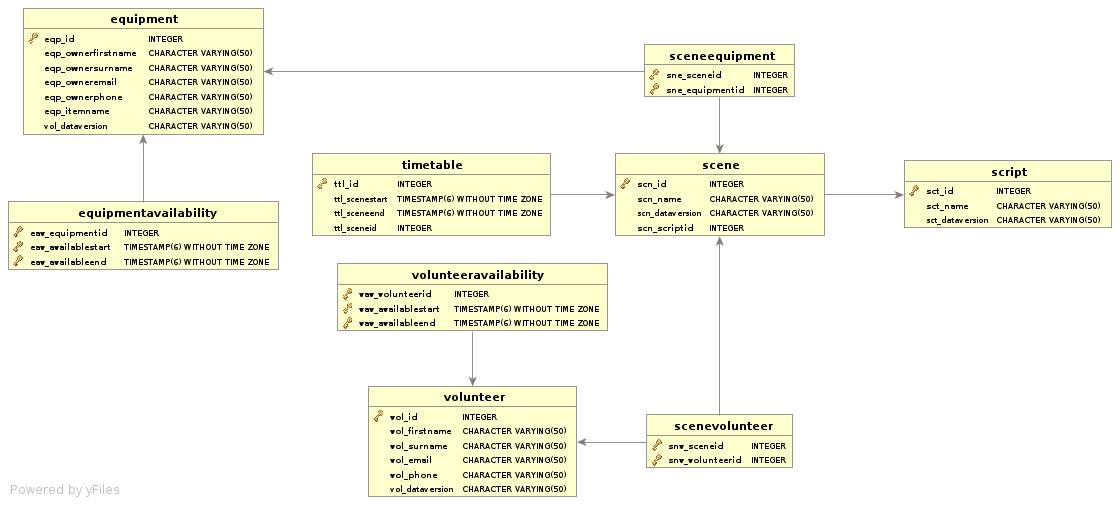
### The business objects:



### Structural Overview:



### The Databse:



### Appendix A: Meeting Minutes

Minutes

Achievatron Unlimited

Sept 23rd: 5:00PM

Present: Iain, John, Ryan, Mitchell

Absent: Matt

Agenda:

1. Conventions of all varieties

2. Milestone 3

3. Others

1.

Coding conventions

Iain expressed his annoyance with the get/set ideas.

We should use the Java style guides:

Classes: SomeClass INterfaces the same

Methods should be verbs, in mixed case with the first letter lowercase,

with the first letter of each internal word capitalized.

Variables: All have meaning, camel case.

0Except loop counters:

vector<widgets> widgetCollection

for (widget currentWidget:widgetCollection)

Unless unavoidable

Git Conventions:

Branches:

Iain has in the past used a 1:1 ratio with issues and branches, but that might not work while we're building

In discussing the conventions it was suggested that we do the web applet as a prototype for the localO

Only other Git things:

git pull origin master: It updates to the most recent version, always do this before starting to work

commit after you're done, but only push if it's tested

We decided on pair programming:

Matt <=> John

Matt=>Ryan

Ryan=>John

Iain<=>Mitchell

We distributed the workload.