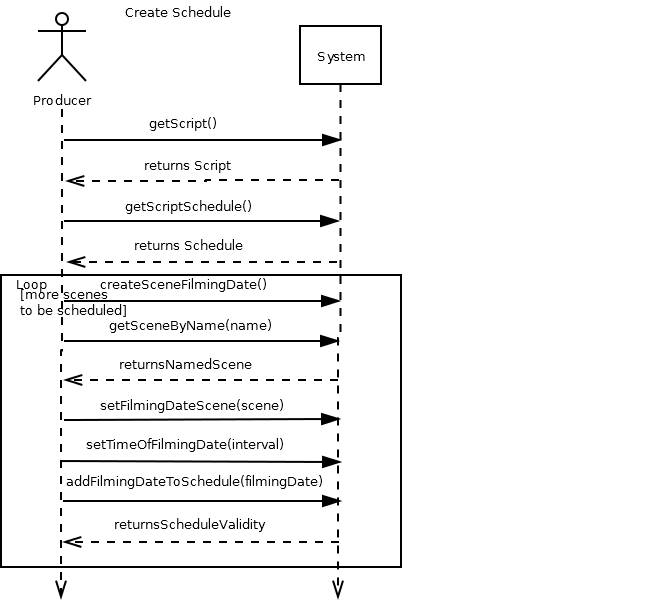
# Milestone 4 – Requirements and Preliminary Design



###### Volunteer Movie Scheduler

### Summary Use Cases

**Use Case: Producer creating the script**

**Actor: Producer**

The clicks the create script button with the intention of putting a new script into production. The producer provides the name of the script, and the system saves it into the database. The producer is then returned to the main menu and given a message indicating success.

**Use Case: Producer creating a scene:**

**Actor: Producer**

The producer looks in the script menu and clicks on the “create a scene” button with the intention of adding a new scene into production. The producer is then asked for the name of that scene, the names of the required volunteers and equipment, and a small description regarding the scene. The system then saves the scene information into the database. The producer is then sent back to the previous menu and given a message indicating whether or not the scene has been successfully added.

**Use Case: Producer removing a scene from a script:**

**Actor: Producer**

The producer wishes to remove a scene from a script and presses the remove button next to the scene he or she wishes to remove. The producer is then prompted with a message that asks if he or she is certain of the decision. If the producer presses yes, they are given a message prompting them of the successful removal of the scene from the script, the system removes the scene information from the database, and the producer is returned to the previous menu. If the producer presses cancel, he or she is returned to the previous screen and the scene is removed from the database.

**Use Case: Removing a script:**

**Actor: Producer**

To a remove a script it must be empty or all scenes within the script must be marked as completed. The producer presses the remove script button with the intention of removing the script. The system then performs a quick check is done to ensure that the above requirements are met. The producer is then presented with a menu that asks whether or not he or she is certain of the decision. If the producer presses yes, system removes the script and all of its scenes from the database. If the producer presses no, the script is not removed.

**Use Case: Completion of a scene:**

**Actor: Producer**

The producer wants to indicate that a scene has been completed. He or she must then click the checkbox next to the scene. The system then removes the scene from consideration when generating a schedule, but keeps it in the database as a reference.

**Use Case: Adding volunteer/equipment to a scene:**

**Actor: Producer**

The producer wishes to edit the requirements of a scene and presses the edit button next to the scene. The producer is then presented with a list of the scenes requirements. The producer must then click the add requirement button to add a required volunteer or piece of equipment to the scene. Once the producer has indicated the volunteer or piece of equipment, the requirement is added to the scenes requirements. The system then performs a conflict check to ensure that the newly added volunteer or piece of equipment is available when that scene is scheduled to be shot. If no conflict is detected, the producer is returned to the previous screen and given a message indicating the successful addition of the requirement. If a conflict is detected, the producer is informed that the scene can no longer be filmed at its scheduled time.

**Use Case: Removing a volunteer/equipment from a scene:**

**Actor: Producer**

The producer wishes to edit the requirements of a scene and presses the edit button next to the scene. The producer is then presented with a list of the scenes requirements. The producer must then click the remove requirement button. The producer is then asked whether or not he or she is certain of her decision. If the producer presses yes, the requirement is removed from the screen and the producer is brought back to the previous menu. If the producer selects no, then the requirement is not removed from the scene.

**Use Case: Producer creating a schedule**

**Actor: Producer**

The producer wants to begin the process of scheduling scenes. The producer opens the schedule tab and views the current schedule at a glance. The scenes that have not yet been scheduled will appear on the left in a list, and the producer will click the scene and assign it to a date, and if the scene can get scheduled there (There's no conflict) then it is.

**Use Case: Conflict Resolution**

**Actor: Producer**

The producer has a conflict, after trying to schedule a scene, so they're brought to another window which has various options for resolving the conflict. Ignore means that the scene will be scheduled anyway, and the producer will be prompted to confirm this. The other basic option will be to contact all those involved in the scene through e-mail.

**Use Case: Producer "Printing" The schedule**

**Actor: Producer**

Once all conflicts have been resolved and every scene schedule the producer wishes to create a tangible document in the form of a spreadsheet or a pdf of the schedule. On the schedule scene, the producer will have a button that says print schedule, which will only be available if the schedule is not in conflict.

**Use Case Name: Add Scene to Schedule**

**Actors: Producer**

The Producer wishes to schedule the date and time which a scene will be filmed. The Producer switches to the 'Scheduling' tab, which displays a list of all the scenes, plus a monthly calendar view centred on the current month, which displays the current shooting schedule. The Producer then Right-Clicks the appropriate date on the calendar and selects 'Schedule Scene' from the context menu which appears. This will open a small dialog window, defaulting to the date which was under focus when the Producer Right-Clicked. The Producer can then select a Scene from a list, and alter the date-time when it will be shot. The Producer can then click 'Okay' which will add the scene to that date in the schedule, or 'Cancel' to back out of the operation. After the scene is added to the schedule, the schedule is then validated to ensure that no conflicts of availability have arisen from the change in the schedule.

**Use Case Name: Remove Scene from Schedule**

**Actors: Producer**

The Producer wishes to remove a scene from its current scheduled time. The Producer switches to the 'Scheduling' tab, which displays a list of all the scenes, plus a monthly calendar view centred on the current month, which displays the current shooting schedule. The Producer then Right-Clicks on the appropriate scheduled scene that they wish to delete and selects 'Delete Scene <Scene\_Name>' from the context menu which appears. A message dialog window then appears asking the Producer to confirm deleting the scene from the schedule. If they click 'Yes' on the dialog then the Scene is removed from the schedule, otherwise no action is taken.

### Fully-Dressed Use Cases

**Use Case Name:** Producer Adds Scene to Schedule

**Scope:** Primary Volunteer Scheduling Application

**Primary Actor:** Producer

**Stakeholders and Interests:**

*Producer*: Wants to be able to schedule scenes to be shot in an appropriate time scale. The producer wants access to all the information on when the scenes can be shot, fitting around the limited availabilities of the volunteers and equipment which the scene requires. The responsibility of the producer is to ensure that the film is fully shot in as short a time period as possible, so having access to information and feedback on the times which scenes can be shot is imperative.

*Volunteer:* Wants to be involved in the filming process at times that suit them. Are invested in the success of the film and as such wish to see the film completed successfully. Each one is giving over their own free time, so wishes to see the film completed in a short a window of time as possible, so as not to have the film unduly interfere with their daily lives.

*Film Company*: Outside of the producer and the volunteers there are also a number of people involved in the creation of the film (writers, executive producers etc.), who might not be listed specifically as volunteers for the purpose of the system, but are still heavily invested in the success of the film.

**Preconditions:**

* The script which the scenes belongs to has been created
* The scenes the producer wishes to schedule have been created and added to the script
* The scenes have not already been shot

**Postconditions:**

* The scenes are added to the schedule at the date/times specified
* The producer is provided with feedback regarding whether or not the scenes can be filmed on the given dates, given the availabilities of the volunteers and equipment they contain

**Summary:**

The Producer wishes to schedule the date and time which the scenes will be filmed. The Producer switches to the 'Scheduling' tab, which displays a list of all the scenes, plus a monthly calendar view centred on the current month. The Producer then Right-Clicks the appropriate date on the calendar and selects 'Schedule Scene' from the context menu which appears. This will open a small dialog window, defaulting to the date which was under focus when the Producer Right-Clicked. The Producer can then select a Scene which has not already been shot from a list, and alter the date-time when it will be shot, if required. The Producer can then click 'Okay' which will add the scene to that date in the schedule, or 'Cancel' to back out of the operation. After the scene is added to the schedule, the schedule is then validated to ensure that no conflicts of availability have arisen from the change in the schedule and displayed in the calendar view. This is then repeated for all scenes that are required to be scheduled until such time as the entire script scenes have been scheduled.

**Main Success Scenario:**

1. The producer starts the system

2. The producer clicks on the 'Scheduling' tab

3. The producer right-click on the date they wish to schedule a scene

4. The dialog window for adding a scene to the schedule appears, with the date defaulted to the date on which the producer right-clicked

5. The producer selects the scene they wish to schedule

6. The producer clicks the 'Okay' button in order to add the scene to the schedule

7. The scene is added to the schedule on the date the producer originally right-clicked on, with feedback on whether this particular time is compatible with the availabilities for the scenes required volunteers and equipment.

8. The producer can then go back to step 3 in order the repeat the process for all the required scenes.

**Extensions:**

1a. the script is not within the system yet:

1. The producer is presented with a dialog on starting the system asking them to enter a name for the script

2. The producer enters the name of the script in the dialog

3a. the producer clicks 'Okay'

1a. the name the user added is empty

1. The system presents an error dialog, explaining that the script name is a required field.

1b the name the user added is valid.

1. The script is added to the system

3b. the producer clicks 'Cancel'

1. The system presents a dialog informing the user that a script is required

2. The system closes

4a. the producer wishes to schedule the current scene for a data different to the date on which they initially right clicked

1. The producer selects a different date from the date control

5a. the scene the producer wishes to schedule is not in the system yet

1. The producer cancels the operation and proceeds to the script tab in order to add the scene they wish to schedule

5b. the scene the producer wishes to schedule cannot be filmed with its current requirements on the selected date

1. The producer is presented with a warning indicating that the scene cannot be shot on this date, however they are still able to add the scene to this date in the schedule

2a. the producer adds the scene to the schedule on this date anyway

1. The scene is added to the schedule, but with a visual warning that its current schedule state means it cannot be shot

2b. the producer selects a different date to schedule the scene to be filmed

5c. the date chosen is earlier than the current date-time:

1. The scene schedule is placed in an invalid state with a relevant warning message: "Scheduled date of filming is before today"

2. The warning message is displayed to the producer

7b. the producer selected a different date from that the one which they initially right-clicked on

1. The scene is added to the date which the producer selected in the date control

7. The producer selected a date which placed the scene schedule in an invalid state (earlier than current date-time)

1. A warning dialog is presented to the producer indicating that the scene schedule cannot be saved because of an invalid state including the error message associated with the scene schedule.

**Special Requirements:**

* Must be error tolerant to losses in database communication

**Frequency of Occurrence:** Initially a prerequisite of creating an itinerary for the filming. Will happen less later, only when some major change in circumstance (a volunteer/equipment availability change that renders the schedule not workable).

**Use Case:** Create and add a scene to the script.

**Scope**: primary volunteer scheduling application

**Level:** user goal

**Primary Actor:** producer

**Stakeholders:**

Producer – wishes to store scenes in the script and be able to access them in an organized way.

**Preconditions:**

* The producer has created a script and is looking at the script menu.
* The script has been created.

**Postconditions:**

* The scene is saved to the script
* The scheduling system is notified of the addition.

**Exceptions:**

* A scene does not contain any requirements.

**Main Success Scenario:**

1. Producer clicks on the “add a scene” button.

2. Producer enters scene information and clicks ok.

3. The scene information is added to the script.

4. The scheduling system is notified of the addition.

5. The producer is returned to the previous menu.

Extensions:

2a. the scene information is invalid.

1. The system displays the error message to the producer.

2. The producer clicks ok and is redirected to the “create and add a scene” menu.

2b. the producer does not wish to save the scene.

1. The producer clicks cancel.

2. The producer is returned to the previous menu.

3a. a scene with that name is already in the script.

1. The producer is presented with both copies.

2. The user responds to the error.

1. The producer wishes to overwrite the scene and clicks overwrite.

2a. the scene in the database is already completed.

1. The producer is notified of the previous scenes completion.

1a. the producer wishes to overwrite anyway and presses overwrite.

2. The old scene info is overwritten with the new scene info.

3. The scheduler is notified of the change.

1b. the producer does not wish to overwrite and clicks cancel.

1. The producer is returned to the “create and add scene” menu.

2b. the scene in the database is not complete.

1. The old scene information is overwritten with the new scene information.

2. The producer is returned to the previous menu.

**Frequency of occurrence:** Continuous at the beginning, sparse during production.

**Use Case:** Producer add requirement to a scene.

* The word requirement will refer to either a volunteer or piece of equipment.

**Scope:** Primary volunteer scheduling application

**Primary Actor:** Producer

**Stakeholder:**

Producer- wants a way of storing scene requirements.

**Precondition:** Producer is creating or editing a scene.

**Postcondition**: The volunteer/equipment is added to the scene.

**Exceptions:**

-The added volunteers and equipment have no time availability.

**Main Success Scenario:**

1. The producer clicks the add requirement button.

2. The producer enters the volunteer/equipment information.

3. The producer clicks ok.

4. The requirement is saved to the scene.

5. The scheduler is notified of a change.

**Extensions:**

2a. the producer does not wish to add the requirement.

1. The producer clicks cancel.

2. The producer is returned to the previous screen.

3a. the requirement information is invalid.

1. The producer is notified of the error.

2. The producer is sent back to the “add a requirement” menu.

**Frequency of occurrence:** Several times for every scene that is created

**Use Case Name:** Resolve conflict by email

The producer has a conflict, after trying to schedule a scene, so they're brought to another window which has various options for resolving the conflict. The most important of these options is to resolve conflicts by emailing the volunteers involved.

**Scope**: Volunteer Film schedule manager

**Level**: User-goal

**Primary Actor:** Producer

**Stakeholders and Interests:**

-Producer: Wants to view all of the information regarding a schedule conflict: The Volunteers and equipment involved, and their respective availabilities, and the reason for the conflict

-Volunteers: Have accurate representation of their availability

-Director: Ensure the system allows the producer makes accurate scheduling decisions.

**Preconditions:** A conflict in the schedule has been identified and associated with a scene

**Postcondition**: The volunteers get emailed about the conflict

**Exceptions:**

There is no conflict in the system

**Main Success Scenario**

1. The producer identifies a conflict and clicks on the conflict resolution button from the schedule

2. A new window comes up that describes the list of conflicts

3. Producer selects a scene in conflict

4. The producer resolves the conflict by sending the volunteers an email

5. The producer is returned to the conflicted scene list.

**Extensions:**

2a. there is no scene in conflict

1. The system notifies the producer

2. The producer is returned to the schedule

3a. the system can suggest a solution or allow the producer to ignore the conflict

**Frequency of occurrence:** Relatively often, as the likelihood of a conflict is high.

**Use Case:** Create Volunteer

**Primary actor:** Volunteer

**Stakeholders and Interests:**

* Volunteer: wants to be able to give their information and availability
* Producer: wants to be able to view and use the information provided
* Other volunteers: want the volunteer to be in the system

**Preconditions:**

Volunteer is identified and asked to fill out form

Postconditions:

* Volunteer has entered all information correctly.
* Volunteer’s information is saved in the database.
* Volunteer has a way to access and modify their data.
* Producer is able to log in and view availability.
* Volunteer’s information is automatically integrated into scheduling process.

**Main success scenario:**

1. Volunteer turns on the computer

2. Volunteer enters their basic information including name, email, and phone.

3. Volunteer enters their availability

4. Volunteer clicks on the OK button

5. The Volunteer’s information is updated in the database.

6. System processes the data.

**Extensions:**

a: Volunteer does not put information in all of the boxes:

1. Error message pops up

2. Volunteer appropriately fills out missed field

3. Volunteer clicks on the OK button

4. System proceeds as normal

b: At any time if the system fails:

1. Do a system reboot

2. Volunteer proceeds to fill out information as normal.

note\* there is not enough information required to back it up while filling out the form

c: Human decides not to volunteer any longer:

1. Human presses cancel.

2. System ignores any information previously entered.

**Frequency of Occurrence:** used N times where N is the number of volunteers.

**Open Issues:**

* Data recovery on crash.
* Should different types of volunteers enter different data?
* Help if the volunteer does not understand something about the forms.

**Use Case Name:** Volunteer Logs in to the Volunteer Availability Application

**Scope:** Secondary Volunteer Availability Application

**Level:** User-goal

**Primary Actor:** Volunteer

**Stakeholders and Interests:**

Volunteer: Wants to be able to access the volunteer availability application in order to review the information they have on file, as well as change contact and availability information.

**Preconditions:**

* Volunteer's credentials exist in the database

**Postconditions:**

* Volunteer has entered valid credentials
* Volunteer is logged in to the system
* Volunteer's information is loaded into the system

**Main Success Scenario:**

1. Volunteer starts the system and is shown the login screen

2. Volunteer enters username (e-mail) and password

3. Volunteer clicks 'Okay'

4. Volunteer's username and password are validated

5. Volunteer is shown the main screen with the volunteer's information loaded from the database

Extensions:

2a. Volunteer no longer wishes to login

1. Volunteer clicks 'Cancel'

2. The system closes

3a. Volunteer did not enter a username (e-mail)

1. The system presents an error dialog, explaining that the username is a required field.

3b. Volunteer did not enter a password

1. The system presents an error dialog, explaining that the password is a required field

4a. Volunteer's username (e-mail) and password combination are not in the system

1. The system presents an error dialog, explaining that the credentials are not valid.

**Frequency**: Occurs once every time Volunteer accesses the volunteer availability application

**Special Requirements:** Connection to the internet

### Supplementary Specifications

**Functionality**

**User Differentiation**

In order to protect the integrity of the system, producers and volunteers will have different privileges, which is achieved by having one main, localized, system with a separate login for the volunteer’s system.

**Modifiability**

Through the main system, the producer can make changes to the script, change the schedule, and access the contact information of all volunteers and equipment. Through the volunteer’s system, the volunteers will only be able to view the schedule and change their own contact information.

**Usability**

**Easy to Navigate**

The project should be easy to understand and navigate. This will be achieved through accurate descriptions of the actions to be performed, and an easy to use interface.

**Easy to Correct**

The project should allow for an easy correction of mistakes. It should be easy to back out of every action before it becomes permanent.

**Consistency**

The user should only be able to create and add volunteers and equipment that don’t already exist within the database (i.e. no duplicates).

**Quick generation**

The generation of the schedule should be as quick as possible and not result in a temporary freeze. The schedule should also attempt to find the schedule that allows for the filming of all of the scenes as quickly as possible.

**Easy to modify**

It should be easy to add, remove, and modify information regarding scenes, volunteers, and equipment.

**Quick conflict detection**

Conflict detection will occur every time something in the system changes; for example, a time availability of a volunteer changes. Any new conflicts should be immediately reported to the producer and suggested solutions should be readily available.

**Error information**

Any error that occurs should provide a sufficient description for the user to be able to resolve the issue.

**Supportability**

**Adaptability**

The system will be compartmentalized to allow for changes to occur without affecting the entire system.

**Future Proof**

The system should be made to allow for increased future functionality. These may include, but are not limited to, multiple scripts, and different teams of volunteers for different scripts.

**Implementation constraints**

**Language Tool**

Achievatron Unlimited is using Java technologies solution. This will facilitate portability, ease of use, support, and modifiability.

**Database Management System**

Achievatron Unlimited Will be using the object relational database management system PostgreSQL to meet the needs of the project. This will allow for a reduction in data redundancy, fast and efficient storage and retrieval of information, efficient organization of data, and a stable foundation for the organization of data

**Legal issues**

**Lost/ stolen/ damaged equipment**

Any loss of, or damage to, equipment should be removed from scheduling and documented to allow the producer to give an accurate report to the equipment owner.

**Application specific rules**

**Scene cooperation**

The volunteers and equipment selected for a scene must all share a time availability, otherwise visual feedback will be provided to indicate that this scene cannot be shot with the current time constraints.

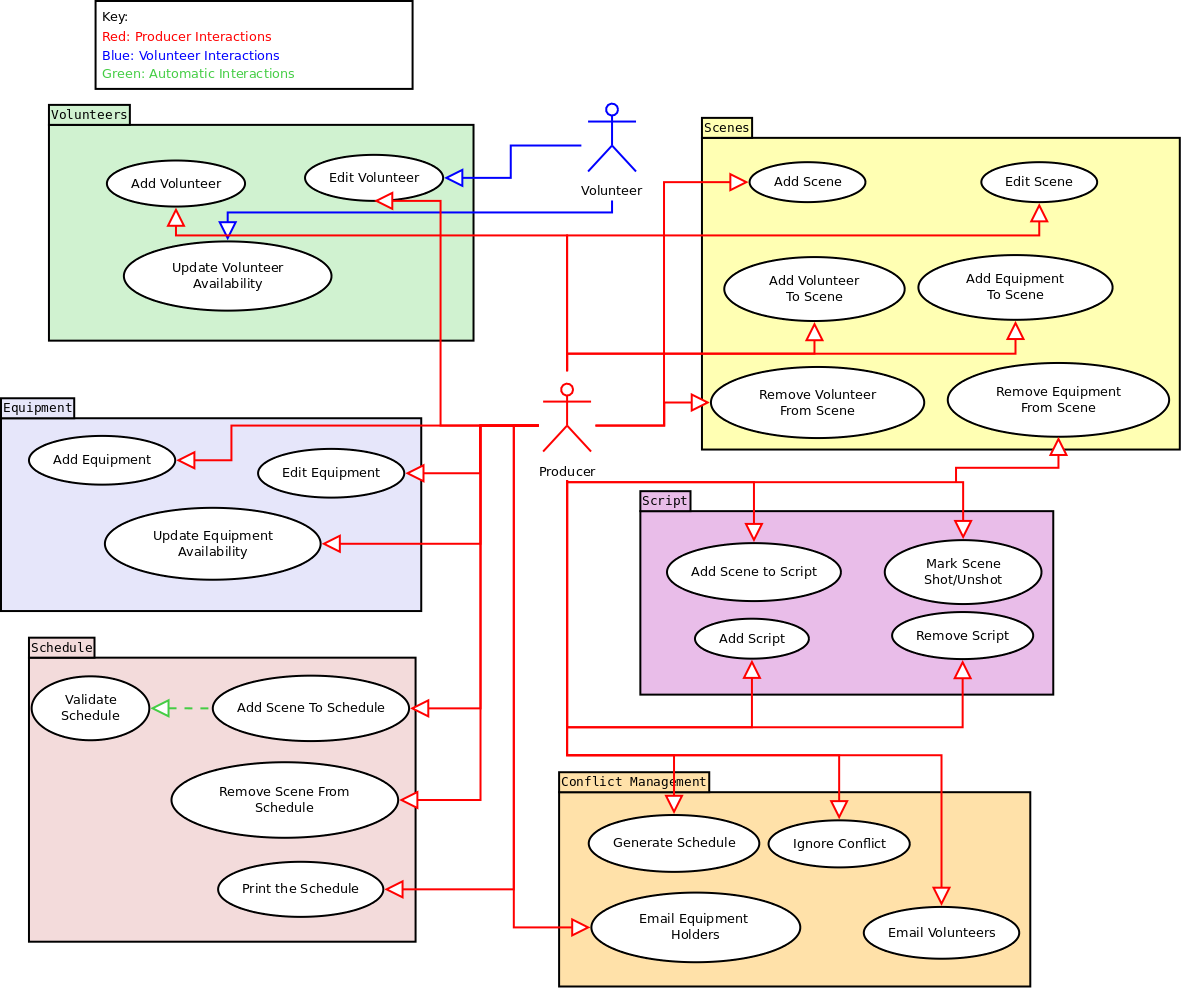
**Information in domain of interest**

**Scheduling**

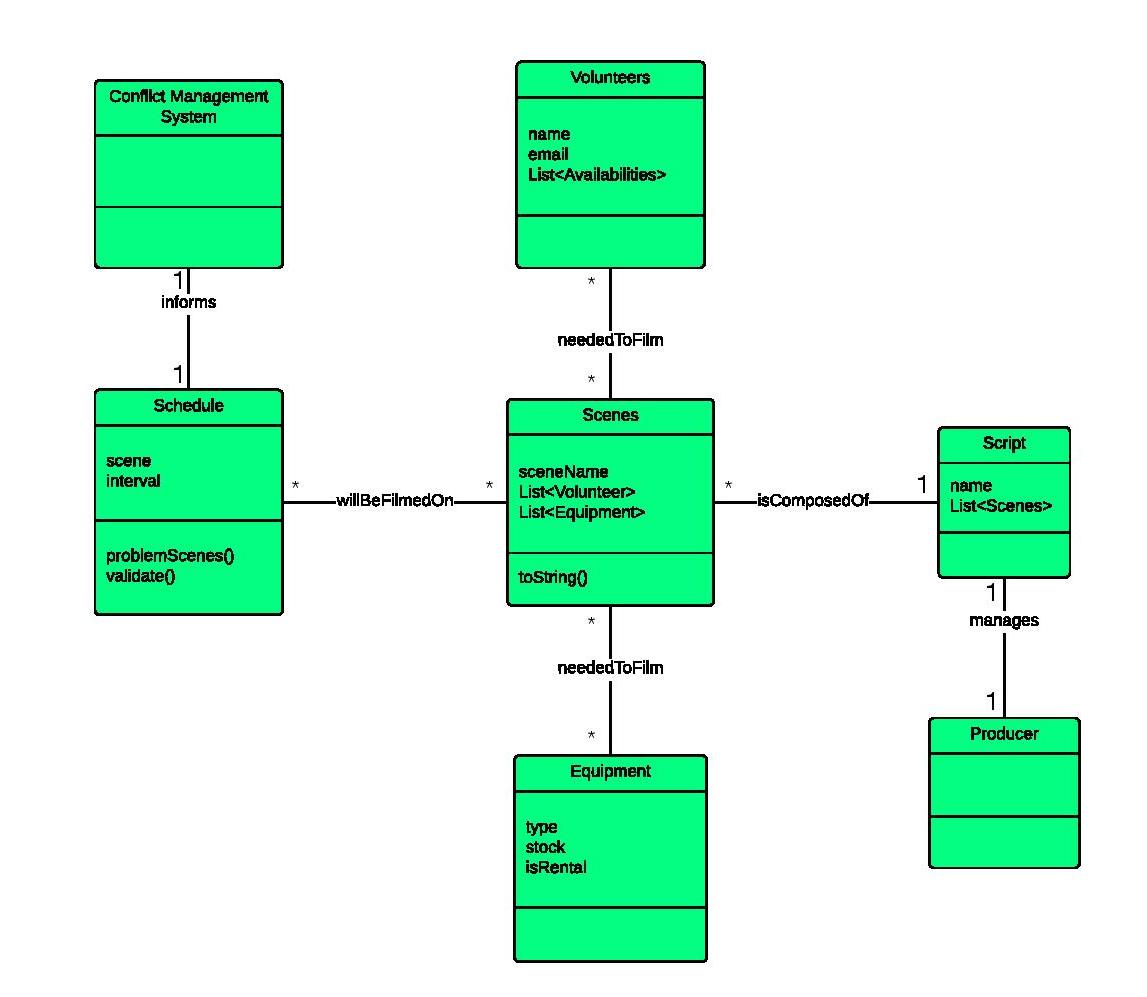
We need to maintain the integrity of the schedule. If a change occurs that results in a conflict with the current schedule, the producer is notified immediately. Also, to increase security and consistency, the producer is the only person allowed to change the equipment availabilities.

The system should be able to handle different date formats as well as both military and standard time.

### Use Case Diagram

****

### Domain Model



### Glossary

**Action ~** Term to describe any class which derives from BaseAction.

**BaseAction ~** Abstract class representing any save or load action for any data that must be persistent between running instances of the system.

**BaseBusinessObject ~** Abstract class from which most of the data objects in the system will be derived, contains mechanisms for validating, and notifying BusinessObjectListeners of changes to its state.

**BusinessObject ~** Term to describe any class which derives from BaseBusinessObject.

**BusinessObjectList ~** Generic container class to hold BusinessObjects, implements the BusinessObjectListener interface.

**BusinessObjectListener ~** Interface which will be implemented by any class which wishes to register itself with a BusinessObject in order to be notified of changes to its state.

**Database ~** Abstract class representing a database in which to store the persistent system data.

**Equipment ~** A class which represents the object required to film a scene.

**JdbcDatabase ~** A class which derives from Database which provides an implementation of the abstract functions in Database using the Jdbc driver.

**SaveSceneFilmingDateAction ~** A class which represents an Action to save a SceneFilmingDate

**Scene ~** A class which represents the section of the film which requires filming. The class has an 1,,n -> 0,,n relationship to Equipment and Volunteers.

**SceneFilmingDate ~** A class which represents the TimeInterval in which a Scene will be filmed. The class has a 1,,1 -> 1,,1 relationship to TimeInterval and a 1,,1 -> 1,,1 relationship to Scene.

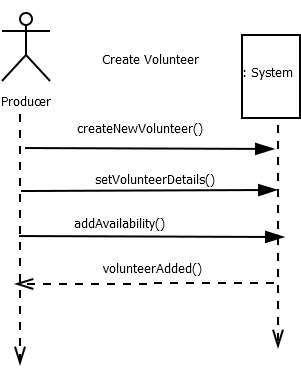
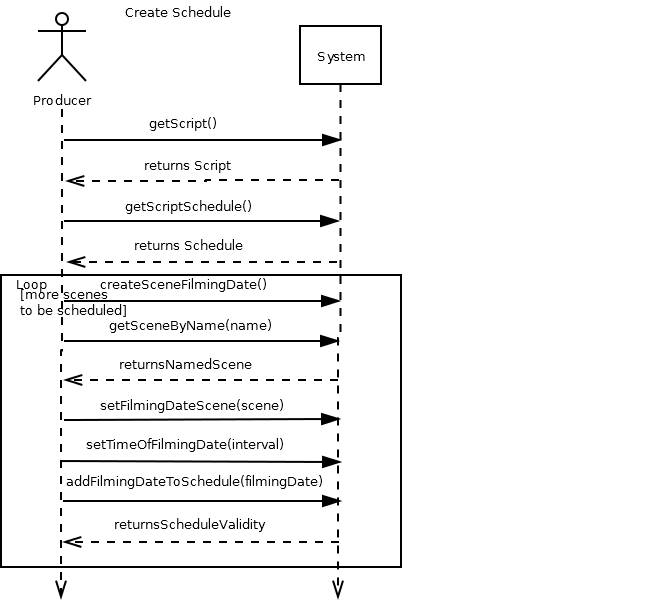
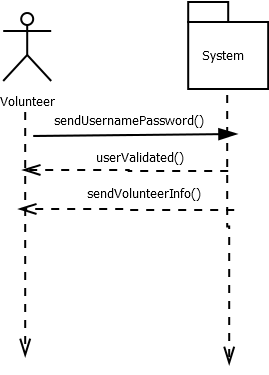
**Script ~** A class which contains all the Scenes which require filming, the Volunteers required to film all the Scenes, the Equipment required to film all the Scenes and the Schedule which details when each Scene will be filmed. The class has a 1,,1 -> 0,,n relationship to Scenes, a 1,,1 -> 0,,n relationship to Volunteers, a 1,,1 -> 0,,n relationship to Equipment and a 1,,1 -> 1,,1 relationship to Schedule

**TimeInterval ~** A class representing a period of time. Has a start and an end. To be valid the start must be <= the end.

**Volunteer ~** A class representing a person who has volunteered their time to be involved in the filming process (eg. camera operators, actors, directors, make-up artists etc).

# System sequence diagrams

# C:\Users\Mitchell\AppData\Local\Microsoft\Windows\INetCache\Content.Word\conflictsysseq.pngC:\Users\Mitchell\AppData\Local\Microsoft\Windows\INetCache\Content.Word\createScene.png



### OPeration contracts

**Operation**: sendEmailtoVolunteer()

**Cross References:** Use case: Resolving conflicts through emails

**Preconditions:**

* The volunteer emails exist
* There is a connection through which the emails can be sent

**Post conditions:**

* The Volunteers have been emailed and properly informed of the conflict.

**Operation:** getScript()

**Use Case:** Create Schedule

**Preconditions:**

* A Script has been initialized for the system to manage

**Postconditions:**

* the Script that the system is managing was returned

**Operation:** getScriptSchedule()

**Use Case:** Create Schedule

**Preconditions:**

* A Script has been created for the system to manage

**Postconditions:**

* The schedule associated with the script was returned

**Operation:** createSceneFilmingDate()

**Use Case:** Create Schedule

**Preconditions:**

* A Script has been created for the system to manage

**Postconditions:**

* A SceneFilmingDate instance sfd was created
* timeInterval in sfd was initialized to NULL
* scene in sfd was initialized to NULL

**Operation:** getSceneByName(name)

**Use Case:** Create Schedule

**Preconditions:**

* A Script has been created for the system to manage
* The scene with the provided name exists in the Script

**Postconditions:**

* The Scene with the provided name was returned

**Operation:** setFilmingDateScene(scene)

**Use Case:** Create Schedule

**Preconditions:**

* The provided scene is not NULL

**Postconditions**:

* The filming date was associated with the provided scene

**Operation**: setTimeOfFilmingDate(interval)

**Use Case**: Create Schedule

**Preconditions**:

* The provided interval is not NULL
* The provided interval is valid (start time < end time)

**Postconditions:**

* The provided interval was associated with the FilmingDate

**Operation:** addFilmingDateToSchedule(filmingDate)

**Use Case:** Create Schedule

**Preconditions:**

* The provided filmingDate is not NULL
* The provided filmingDate is associated with a scene
* The provided filmingDate is associated with an interval

**Postconditions:**

* The provided filmingDate was associated with the schedule
* The valid state of the schedule was returned

**Operation:** createNewVolunteer()

**Use Case:** Create Volunteer

**Preconditions:**

* There is a volunteer to create
* The volunteer form is open
* The provided email, name, and phone number are filled in correctly

**Postconditions:**

* A new volunteer is created and added to the database
* The database is in a valid state
* The volunteer window is set invisible and disposed of properly

**Operation:** addAvailability()

**Use Case:** Create Volunteer

**Preconditions:**

* A volunteer has been previously created properly
* The provided list of availabilities is not NULL
* The volunteer properly adds their availabilities to the list

**Postconditions:**

* A list of the volunteer’s availabilities is added to the volunteer in the database
* The database is left in a valid state

**Operation:** sendUsernamePassword()

**Use Case:** Volunteer Logs in to the Volunteer Availability Application

**Preconditions:**

* The login menu is open
* The username and password fields are filled in correctly

**Postconditions:**

* The username and password are sent to be validated against the database

**Operation**: userValidated()

**Use Case:** Volunteer Logs in to the Volunteer Availability Application

**Preconditions:**

* The user's username and password have been tested against the database

**Postconditions:**

* The user was informed of their successfulness logging into the application.

**Operation:** sendVolunteerInfo(email)

**Use Case:** Volunteer Logs in to the Volunteer Availability Application

**Preconditions:**

* The user has been validated successfully into the application

**Postconditions:**

* The Volunteer with the associated email was returned
* The MainMenu was initialized and displayed with the associated Volunteer
* The LoginMenu was set to invisible and disposed of properly

### project planning

Our previous project planning diagram, in an updated form:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | Complete |
| Create example databases | Iain | 1h | .5h | Oct 1st | Complete |
| Create Buisness Objects | All | 3.5h | 5h | Oct 4th | Completed |
| UML Modelling of system | All | 2h | 2.5h | Oct 15th | Completed |
| Decide on Conventions | All | 1h | 1h | Oct 1st | Completed |
| Code: Volunteer/equipment Availability | John | 8h | 10h | Oct 31st(M4) (Primary Success) | Completed |
| Code: Scene Requirements | Ryan | 8h |  | Mid Nov | In Progress |
| Code: Schedule Generation | Iain | 10h |  | Mid Nov | In Progress |
| Code: Conflict Resolution | Mitchell | 9h |  | Mid Nov | In progress |
| Code: Applet Prototype | Matt | 7h | 10h | Oct 31st  (Primary Success) |  |
| Usability Testing | ALL |  |  |  | Not Started |
| Code: Central UI | Iain | 2h | 5h | Oct 31st | Completed |
| 4. Use cases | Ryan | 3h | 2h | Oct 31st | Completed |
| 4. Specifications |  | 2h | 1h | “ | Completed |
| 4. Use Case Diagram | Iain | 1h | 1.5h | “ | Completed |
| 4. Domain Model | John | 1.5h | 1.5h | “ | Completed |
| 4. Glossary | Matt | 1h | 2 | “ | Completed |
| 4. System Sequence Diagram | All | 4h | 6h | “ | Completed |
| 4. Contracts | All | 2h | 4h | “ | completed |
| 4. Implementation | All | (Derived from above) | 20h | “ | Completed |
| 4. Project Plan | Mitchell | 1h | 1h | “ | Completed |
| 4. Meeting Minutes | Mitchell | .25h | .25h | “ | Not Started |
| 4. Ensure proper format for hand in | Mitchell | 1h | 1h | “ | Not Started |

% Contribution

Iain – 40%

John – 10%

Mitchell – 15 %

Ryan -- 20%

Matt 15%

Milestone 5 Requirements:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirement | Who? | Expected time | Real time | Completion Date | Completed? |
| System Sequence revaluations | All | 2h |  | Oct 14th | Not Started |
| Class Diagram | All |  |  |  | Not Started |
| Implementation | All |  |  |  | Partially Completed |
| Project Plan | Mitchell |  |  |  | Partially |
|  |  |  |  |  |  |

Implementation expectations:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Implementation Goal | Who’s completed it for their section/Responsible | Expected time | Actual Time | Completion date | Completed? |
| Major Success Scenario | Matt/John | 60-70h total |  | Oct 14th | Partially Started |
| UI | Iain, Mitchell | 20h total |  | “ | Partially Completed |
| Naming Conventions |  | 2h |  | “ | Discussed |
| Comments |  | 5h |  | “ | “ |
| Consistent Style |  | 4h |  | “ | “ |
| User Manual | Matt | 2h |  | “ | “ |
| Git Log | Iain | 1h |  | “ | “ |

### appendix

Meeting minutes oct 14th

Present: All

In this meeting we discussed the creation of the system sequence diagrams, and drew them out on the board to match their fully dressed use cases

Meeting Minutes 21st October 2014

Present: Iain, Ryan, John Mitchell

Not Present: Matt

In this meeting Iain gave us an overview of the use of actions and business objects within the scope of our application as well as a quick overview of establishing a database connection for John and Mitchell since they’re not in databases

Meeting Minutes 27th October 2014

Present: Matt, Iain, Ryan, John

Not Present: Mitchell

Item 1 - The sprint for milestone 4

Documentation

Iain is going to update his fully dressed use case to be inline with the form specified in lecture/text book.

Matt is going to work on his fully dressed use case today

Ryan’s fully dressed use case is already in line

Matt is going to work on the user manual, talked about having details on requirements for the system (JDBCDiver for postgres, Java version 1.8)

Already detailed the Domain Diagram in a flash meeting on Friday, just needs drawing up properly

System Sequence Diagrams. Because operational contracts is reliant on the completion of we need to complete these first. Considered that it would be a better idea for each of us to do these for a fully dressed use case, and take it off mitch’s shoulders.

John has his Fully Dressed use case done, but still has to do his summary use case

Glossary. Talked about maybe mitch doing this if we decide to lighten his load otherwise

Implementation

Success Scenarios: Creating the script is already done. Adding a volunteer will be the second primary success scenario.

Skeleton Implementation: Have function stubs for all the mentioned operation which check for the contract preconditions.

Meeting with group 8

Got an extension!

Further on with implementation than document, slightly. Are meeting on Friday. Have real world potential test client lined up

Use Cases: Login and insert med have been identified as the critical success scenarios.

Risks/Challenges: Getting the simulator for IOS working for development. Risks are the security, mentioned they’re double hash and salting the sensitive data, and sending in packets small enough to not be able to identify the data if not part of the session.