

**UpStage Project Proposal**

Date 25/03/2014

Version Number 5.0

**Team Members**  
Nikos Phillips

Vanessa Henderson

James Williams

Xiangyu Chen  
Takuma Sato

**Supervisor**

Anne Philpott

**Clients**  
Vicki Smith

Helen Varley Jamieson

Table of Contents

1. Project and Team………………………………………………….………………...………5

* 1. Clients
  2. Supervisor
  3. Continuing Team Members
  4. New Team Member

2. Team Roles…………………………………………………………………….………..……6

3. Terms of Reference………………………………………………………………………….7

3.1 UpStage Background Information

3.2 Use in the Community

3.3 Rationale for this Project

3.3.1. Age of Product

3.3.2. State of Code

3.3.3. Accessibility and Portability

4. Objectives……………………………………………………….…………………….………9

4.1. Finish Version 3

4.2. Design a New UpStage

4.2.1. Accessbility

4.2.2. Portability

5 Scope…………………………………………………………………………………………10

5.1. End of Production for Version 3

5.1.1. Tickets

5.2. Investigating New Software

5.2.1. Overview

5.2.2. Time Constraint

5.2.3. Tasks

6 Skills and Knowledge……………………………………………………………….………11

6.1. Getting Accustomed to the Software

6.2. Languages

6.3. Understanding Old Work

6.4. Skills Involved

6.4.1. Personal

6.4.2. Professional

6.4.3. Technical

7 Activities………………………………………………………….…………………..………14

7.1. First Three Months

7.2. Four to Six Months

8 Approach and Methodology…………………………………………………..……………15

8.1. Adaptable Development

8.2. Pair Programming

9 Quality Assurance…………………………………………………………..………….……15

9.1. Pair Programming

9.2. Mid-Project Review

9.3. Supervisor Meetings/Client Meetings

9.4. Testing

10 Quality Control……………………………………………………………………...………16

10.1. Coding Standards

10.2. Version Control

10.3. Issues Log

10.4. Risk Register

11 Deliverables………………………………………………………………………...………18

12 Costs……………………………………………………………………………………...…18

13 Schedule and Plan…………………………………………………………………………19

13.1. Overall View of the Schedule

13.2. Sprint One – Semester 1, 2014

13.3. Sprint Two – Semester 1, 2014

13.4. Semester Two, 2014

13.4.1. Approach

13.4.2. Priority

13.4.3. Deliverable

14 Team Collaboration and Communication……………………………………………..…23

14.1. Team Norms

14.2. Communication Plan

15 Glossary ……………………………………………………………………………………24

16 References……………………………………………………………………….…………25

17 Version History……………………………………………………………………..………26

18 Disclaimer……………………………………………………………………..……………27

Project and Team

**Project Name: UpStage**

**Clients**

Helen Varley Jamieson

Artist and Original UpStage Project Coordinator/Founder

Wellington/Europe

helen@creative\_catalyst.com

Vicki Smith

Artist, Educator and Original UpStage Founder

[vicki@UpStage.org.nz](mailto:vicki@UpStage.org.nz)

**Supervisor**

Anne Philpott

Senior Lecturer

School of Computing and Mathematical Sciences

(09) 921-9999 ext 5216

[anne.philpott@aut.ac.nz](mailto:anne.philpott@aut.ac.nz)

**Continuing Team Members**

Name: Vanessa Henderson

Major(s): Software Development & Computer Science

Email: nessa4@gmail.com

Mobile: 0226521234

Name: Nikos Phillips

Major(s): Software Development?

Email: nphillipskc@gmail.com

Mobile: 021401070

**New Members**

Name: James Williams

Major(s): Software Development & Computer Science

Email: jimiwillz@gmail.com

Mobile: 0212531706

Name: Takuma Sato

Major(s): Software Development & Computer Science

Email: mambo69vn@gmail.com

Mobile: 0221695634

Name: Xiangyu Chen

Major(s): Software Development

Email: cxyyouyou@gmail.com

Mobile: 02102778858

Team Roles:

|  |  |
| --- | --- |
| Project Leader | Vanessa |
| Progress Tracker | Nikos and Takuma |
| Risk Manager | Takuma |
| Communication Manager | Vanessa and James |
| Server Administrator/Technical Manager | Nikos and Xiangyu |
| Scrum Master | Will change with each sprint |
| Software Developer | Everyone |

Terms of reference

**UpStage Background Information**

UpStage is an ongoing open-source, web-based application used for live interactive online performances, building on the basic concept of online chat functionality and adapting it to be used in creative performances. UpStage can allow multiple users to talk by typing but also use video streaming, images, text to speech voices, music and sound effects to make their performances more vivid.

There are two kinds of users for UpStage. “Players” can control avatars on the stage (the main part of the screen) and utilise all the other features listed above. “Audience members” can view the stage but can only interact with others via chat. Anyone who joins the stage without logging into an account will automatically be assigned as an audience member. Players will have been assigned as a Player to that stage prior to the performance.

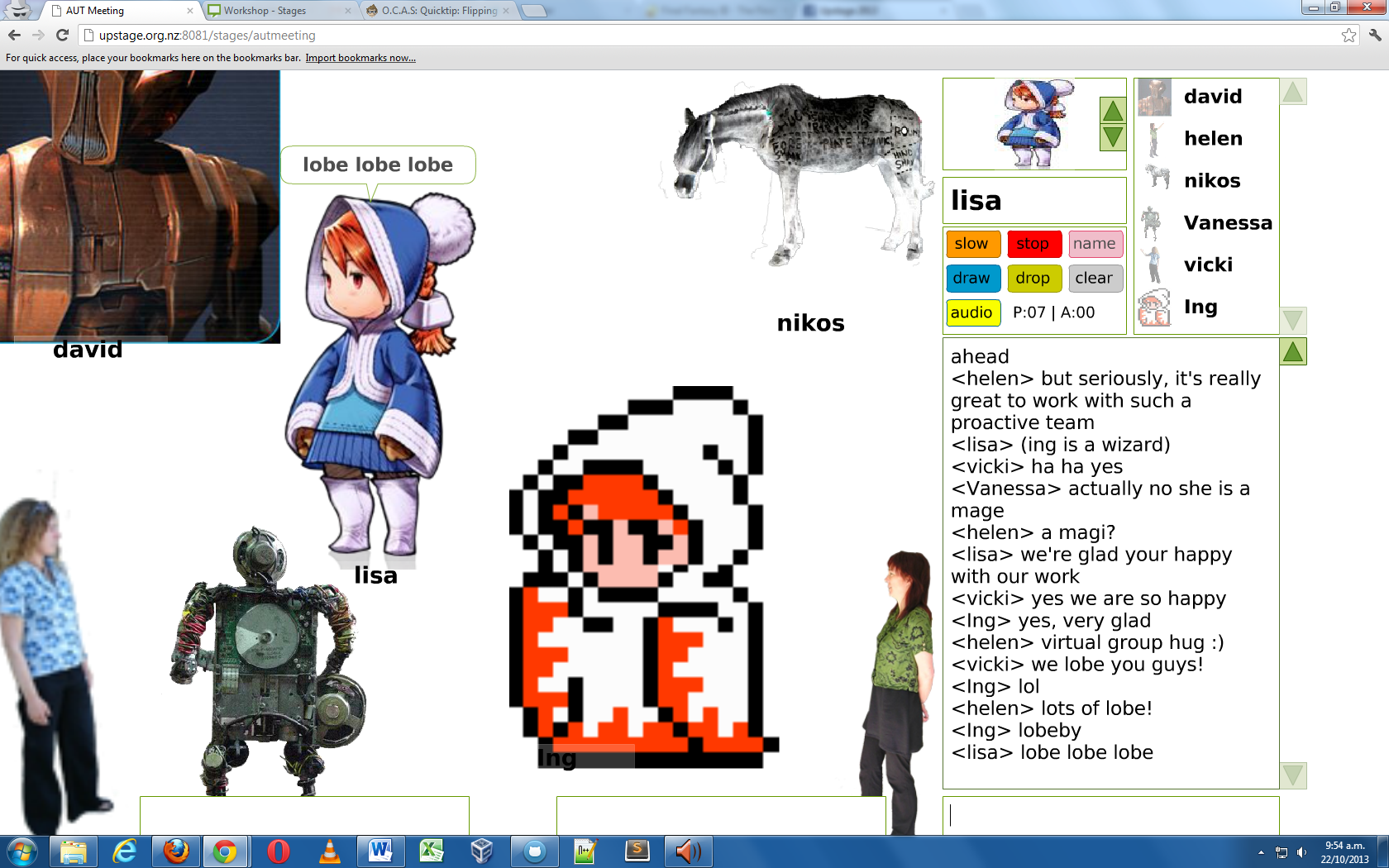


Image 1. UpStage Performance Example (AUT UpStage Team Archive, 2013)

**Use in the community**

UpStage has been used by artists, schools, universities and community groups since its launch in 2004. AUT has been involved with the development of it since 2006. AUT has released Versions 2.0 and 2.1 to the public. The current version accessible on the internet is Version 3.0 which is still in the Beta stage. An example of UpStage being used in the community is by one of our clients, Vicki, who uses it for educational purposes. Some shows that have been performed since the start of the current semester are Balloon and Flood. Flood showed an educational aspect of UpStage, while Balloon demonstrated artistic use of the program.

Rationale for this Project

**Product Age**

There was a two day meeting held at the beginning of this year to celebrate the 10th Anniversary of the software. This meeting reviewed the plans for UpStage’s future. It was decided that development of UpStage will stop after version 3 and a replacement for UpStage will need to be researched, planned and developed. UpStage is now 10 years old and as an open source project, it has been worked on by many different people.

**State of Code**

It has reached the point where the code has become extremely difficult to work with and maintain as well as test. There is a lack of inbuilt automated testing framework to run test suites. Due to a lack of having an automated test suite it takes approximately 40 hours to test and bugs are not easily found. It has become apparent there are some bugs in the current version which are hard to replicate in New Zealand. Some examples of these bugs are voices stuttering, endless voice loops, avatars being switched. Some of the original code is undocumented and does not follow naming conventions. The state of the legacy code makes it very hard to go back and make enhancements or fix bugs.

**Accessibility and Portability**

Another factor driving change is accessibility/portability. The definition of portability has changed since the first version of UpStage was released. By today’s standards to be accessible and portable the software needs to be available for mobile devices such as Android and iOS as well as be accessible from a desktop. There is a serious need for a complete redesign of the software to keep it up to date and relevant for the modern user, and to enable it to be vastly more extensible and maintainable. While redesigning UpStage, the team will be able to research allowing UpStage to be used from a mobile device as well as from a desktop.

Objectives

**Finish Version 3**

The first objective of this semester is to get Version 3 of UpStage finished. This means performing essential bug fixes. These bugs have been added to the issues log on GitHub and tagged as high priority.

Designing a new UpStage

While we work on finishing Version 3, part of the team will begin research into possible avenues of development for a new UpStage product. The part of the team working on that will determine what the new version of UpStage needs to be, how it can improve on the original version, and what it will be limited by in terms of available technology. The team will produce prototypes to demonstrate proof of concept for new ideas. This could broaden the potential user base of UpStage by being deployed for mobile devices, while at the same time offering a completely refreshed and new browser interface for computer users.

Martin Eisenbarth is a developer who has written his PhD thesis based around UpStage and has contributed to the project in the past. Martin has been working on possible extensions for UpStage. The AUT team will contact him in regards to if he is willing to contribute to our project this semester. It could be a good strategy to get help from outside our team to work on new ideas together.

**Accessibility**

A main priority with UpStage has always been for it to be as accessible as possible. This has been achieved by implementing the software entirely as a browser based web application, so that nothing needs to be downloaded and installed before it can be used. While investigating new software the team will keep this in mind at all times.

**Portability**

It has become apparent that the industry standard for deploying applications on mobile devices is by making them available from the mobile OS vendor’s App Store. This goes against the philosophy of users not having to download anything in order to access UpStage. The team will keep our minds open to all possible solutions so as to find the most suitable one for the project.

Scope

End of production for Version 3

The team will complete Version 3 of UpStage before the end of this semester. This will mark the end of the current version of UpStage’s lifespan. The software will continue to be maintained by AUT teams only on a critical bug fix basis, until a new version is fully developed and ready to completely replace it.

**Tickets**

Currently there are 9 outstanding high, and 15 medium priority issues that need attention. Some of these require minimum effort, there are 5 tickets in our first sprint with a value of 1 user story (1 user story = 1 hour). There are 4 tickets worth 10 or more story points, and two worth 5.

Investigating new software

**Overview**

The investigation into new development areas and possibilities will continue into Semester 2, 2014. The investigation will likely continue into 2015. The deliverables from this will be ideas, suggestions and any prototypes that give examples of UpStage utilising modern technology. This will give the clients and future students an idea of what the modern version of UpStage could eventually become.

**Time Constraint**

Due to time constraints for the team working on the project in this semester the features to be included in a new version will not mirror all the current available features in the existing version of UpStage. The team will work towards making a complete package with at least all the functionality of the current version. However, this will likely take place after the current team has left the project and will be carried out by new members.

**Tasks**

The first task will be identifying “must have” functionality from the existing version of UpStage that we will need to include in the new version, this will likely be things such as the use of chat and avatars. We will also identify non-essential features that would be “nice” to include, identify testing frameworks we would want the software to be tested on, identify programming languages and frameworks to be developed in, and how the software could accommodate mobile users.

Skills and Knowledge

**Getting accustomed to the software**

The new members need to learn what UpStage is, how it is used and what it is useful for. We have attended several cyber performances as an audience member and now have a basic understanding of the customer-base. To ensure we have good understanding we will keep attending cyber performance when possible.

**Languages**

The team will need to become more familiar with the coding languages Python, ActionScript, JavaScript as well as HTML and CSS. We also need to learn/refresh our knowledge of Linux/Debian commands that enable us to run the server from Debian, and how servers operate in general. This knowledge should be gained through reading documentation such as online manuals, and frequent pair programming with existing members to ensure we meet the coding standards.

**Understanding Old Work**

We will need to familiarise ourselves with the existing code and learn the architecture so we will be able to track down where the existing bugs actually occur in the code. The existing team members are already familiar with the current version of the program, its code, and how to operate the server it runs from. The new members will ask the continuing members if and when they don’t know something about what they are working on.

Skills Involved

Personal

**Learning**

Learning and understanding the existing work. Each team member will investigate and learn the new programming languages which will be used in developing the software.

**Communication Skills**

The team will demonstrate communication skills via face to face meetings, emails, online meetings and discussion with the team, supervisor and clients to ensure the project is going the right way.

**Teamwork**

We will work as a team by holding team meetings, scrum meetings, retrospective meetings, online discussions within our facebook group, and pair programming. Actions performed independently for the project will be reported to the team. This way, we will work effectively as a team.

**Attitude**

All team members will have a positive attitude for working together and treat other members with respect

**Self Management**

Team members will have to manage themselves, their work and their time well to work effectively. Each member will make sure they do their best to contribute to the team.

**Problem Solving**

Each team member will be able to deal with problems and face challenges. Members will try to figure it out by themselves first so that it will promote the ability of solving problem and save other member’s time.

**Professional**

**Time Management**

Team member will finish assigned work on time. Members will arrive to meetings and workshops on time.

**Project Management**

In order to manage the project appropriately, firstly all the members will form team norms, produce and maintain a realistic schedule, track progress in terms of the plan, and devote ourselves to work on the project.

**Project Tracking**

Tasks, meeting minutes, and time spent on work will be recorded in documents for future reference and evidence.

**Technical**

* Python 2
* HTML and xHTML
* JavaScript
* ActionScript 2
* CSS
* Linux/Debian OS
* Terminal and Shell
* GitHub (for code)
* Subversion (for documentation)

Activities

First Three Months

* Client meetings will be held every 2 weeks with either Vicki or Helen and once a month with both Vicki and Helen. The schedule is to be confirmed.
* Supervisor meeting will be held weekly with Anne unless in development sprints, starting in week 5 of the semester.
* SCRUM meeting online each Saturday and physical each Tuesday
* Carry out Sprint 1
* Scrum Meetings
* Pair programming
* Fix high priority bugs
* Acceptance and regression testing
* Retrospective Sprint meeting
* Sprint 2 planning
* Planning of investigation of new UpStage
* Carry out Sprint 2
* Fix high & medium priority bugs
* Research languages/frameworks/technologies for new UpStage
* Produce reports on findings
* Produce prototypes providing proof of concept

Four to six months

* Mid project review (and proposal for new members)
* Familiarise new team members with tools and project
* Maintain UpStage version 3
* Perform essential bug fixes
* Research applied language/framework use for new UpStage
* Develop prototypes (further if successful. Begin new ones if not)
* Poster presentation

The Future

- *"I never think about the future - it comes soon enough."* --Albert Einstein

* The UpStage project will continue to be worked on by new AUT team members after we have left.

Approach and Methodology

**Adaptable Development**

The AUT UpStage team has decided that Scrum (the agile methodology) will suit the project. “This approach is applicable to small teams developing projects characterised by often changing requirements, tight schedules, and high quality demands” (Ilieva, Ivanov, & Stefanova, 2004). UpStage is user-driven software that is developed and then reviewed in iterations. The team will work in sprints. These will involve pre and post meetings and scrum meetings which are brief and just go over what each person has been working on, what they will be working on next and what, if anything, is stopping them from making progress.

**Pair Programming**

This will be the most efficient way to continue making progress on the work and also bringing the new team members up to speed so they will be able to repeat the process when joined by new members in the next semester. According to Bipp, Lepper et al (2008), “pair programming benefits development the most when one programmer is senior programmer and the other is junior”.

Quality Assurance

**Pair Programming**

There is a coder and a watcher. The person watching is able to spot mistakes such as typos and syntax errors as the other developer codes. This means the watcher can point them out and get them fixed as soon as they occur, leaving less things to debug later on. This will be the first stage of quality assurance when pair programming is put into practice.

**Mid-Project Review**

The mid project review will be a major review of the project and the quality of the work being produced. This will be a check halfway through the time each student will spend on this ongoing project. The review will check to see if enough time has been spent on the project by each student, if the workload isn’t too high for the team, if the work is actually still doable and if it is likely that the team will be capable of finishing the project in the specified timeframe.

**Supervisor Meetings/Client meetings**

Regular meetings with the clients and supervisor will ensure that the project does not stray from where it needs to be headed. Any thoughts, considerations or queries will be put to the clients and supervisor to ensure the team knows exactly what is wanted of them and the project. In the same way the clients and supervisor will be able to check that the team is moving the project forward in the most logical manner.

**Testing**

At the end of each sprint, the software will be regression tested to assure new bugs weren’t introduced and to confirm the functionality requested has been added to the program.

Quality Control

**Coding Standards**

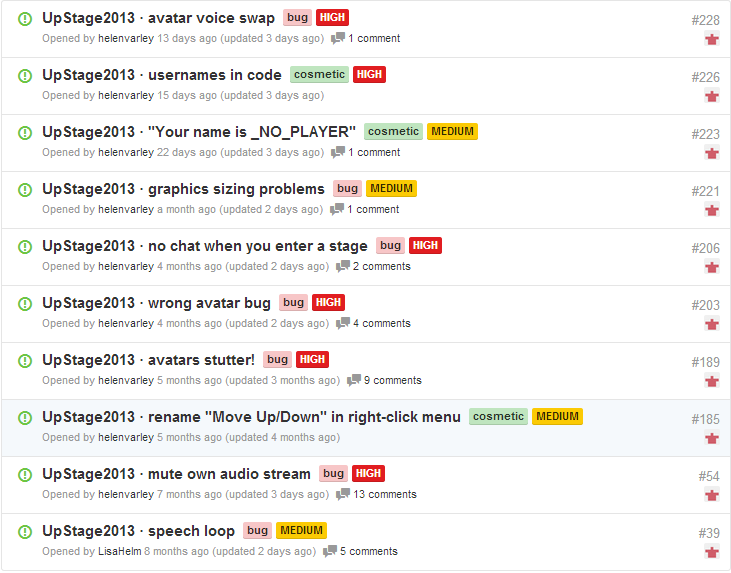
Pair programming for keeping standards. At the end of each Sprint a regression test will be performed which will cover all acceptance tests for the software. This will ensure new bugs aren’t introduced as well as check that the user stories have been completed successfully.

**Version Control**

The project’s code is controlled by and stored on GitHub. It is open source and can be accessed on GitHub by others (e.g. the clients), not just the AUT team.

**Issue Log**

There is an Issues Log on GitHub being used to log any issues or bugs and defects found in the code. The issue is logged along with importance/priority, founder, date, and type of issue.



A sample of the issues logged on GitHub

**Risk Register**

The risk register document logs any risks that may arise during the project that could hinder the team’s progress with the project. Any new risks are added to the register along with possible solutions to overcome them.

Deliverables

* Project Proposal
* UpStage Software - Version 3
* Research reports
* Prototypes
* Project Portfolio
* Poster

Costs

Client and Supervisor Costs

* Time spent on meetings, assessing work and giving feedback, responding to emails.

AUT Costs

* Resources - Dedicated PCs - Server
* Internet access
* Printer
* Project room
* Proprietary software

AUT students

* Time spent attending workshops, meetings, and working on project, 150 hours per student 15 point paper, 300 hours per student per 30 point paper.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Individual hours to contribute | Cumulative hours | Weekly contribution (12 weeks) |
| Takuma | 150 | 150 | 12.5 |
| Xiangyu | 150 | 300 | 12.5 |
| James | 150 | 450 | 12.5 |
| Vanessa | 300 | 750 | 25 |
| Nikos | 300 | 1050 | 25 |

Schedule and Plan

As a team we decided to have two sprints this semester. This will cover all essential bug fixes that need to be completed for the release of version 3. Each sprint will have a fixed duration of two weeks, as it will allow us to manage time constraints.

The unit for measuring ‘Effort’ are story points and each story point represents an hour’s worth of work. Tasks will be assigned to sprints based on the figure shown below.

Table of team Effort

|  |  |
| --- | --- |
| Points per Day | 12.5 Points |
| Point per Sprint | 175 Points |

**Overall view of the schedule**

The figure below is an overview of the schedule for this semester. There will be 2 sprints from 27/03/2014 to 24/04/2014 and we aim to resolve all high priority tickets in sprint 1 and all essential medium tickets by the end of sprint 2. After sprint 1, part of the team will begin researching tools for the new software while the rest of the team will enter sprint 2. By having a parallel process, transition from sprint 2 to investigation of new software can be done in a smooth manner without a period of downtime in between. This will prevent a gap between sprint 2 and the research for the new UpStage with little to know productivity

Key

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | Online SCRUM meeting | Physical SCRUM meeting | Acceptance and regression testing |
| Planning of investigation | Research for new Upstage | Produce Report on finding | Produce prototypes |

Figure 1. March 27 - April14th

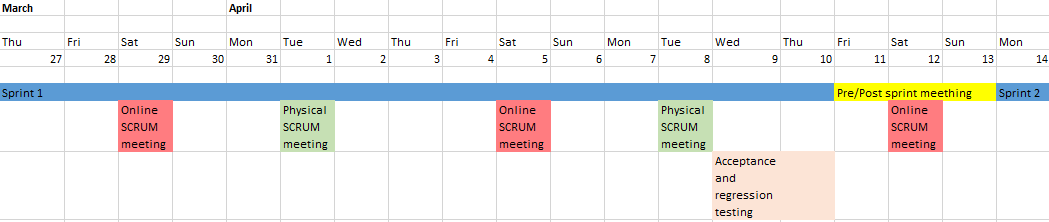


Figure 2. April 14 - April 30th

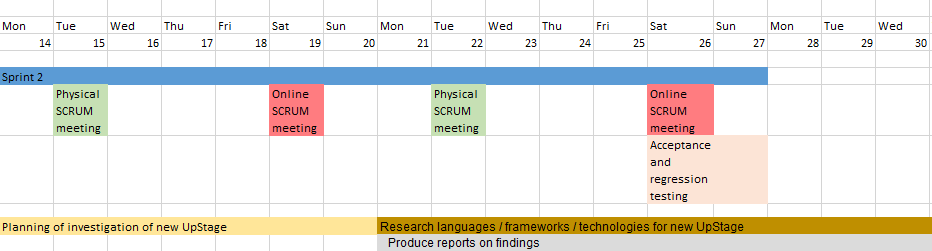
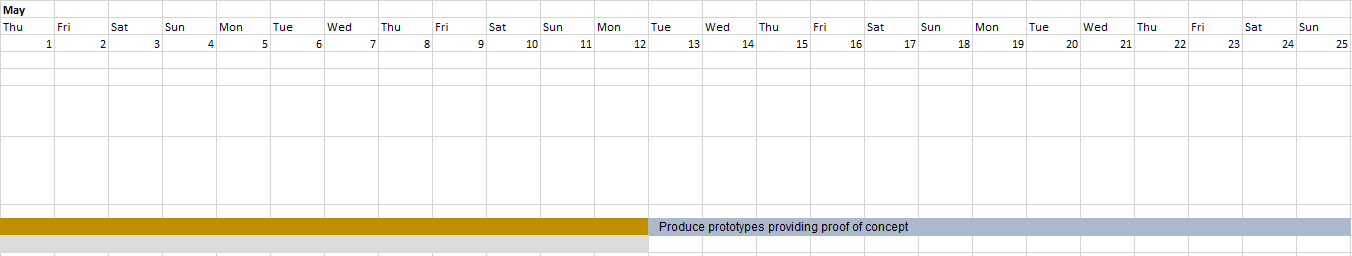
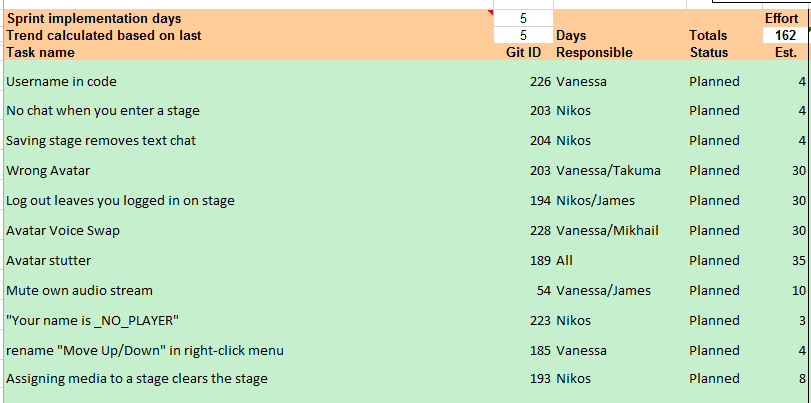


Figure 3. May 1st - May 25th



Sprint 1, Semester 1, 2014 - Resolve high priority issues

Sprint 1 will be used to debug major bugs with high priority assigned to AUT Upstage team on Git-hub. New functionality will not be added during sprint 1. The goal of this semester is to create a stable Upstage version 3 as discussed with the client.



The estimated effort for these stories is intentionally below our 175 points allocated to each sprint to cover unforeseen issues or problems that may arise. Our time will also be spent attending class, meetings and documenting our work. Regression testing at the end of each sprint will take roughly 40 team effort points.

Sprint 2 Semester 1, 2014 - Stabilize version 3

Sprint 2 will be used to resolve medium priority issues along with any carried over tasks from sprint 1. There will be 27 medium priority issues (9 tagged as bugs) remaining. These will be reviewed in a pre-sprint meeting in order to identify which of them are essential bugs/issues which need to be patched for version 3 and to also identify how much effort is required to complete them.

Depending on the result of sprint 1, we will have 2 or 3 team mates dropping out to work on new software investigation. We plan to provide a proof of concept for a new version of UpStage in the last 3 weeks via a functioning prototype. The specifics of the prototype are yet to be determined. The prototype will demonstrate some variety of sought after functionality from UpStage, however, it will be an all new program.

Semester 2 – 2014

The remaining three members will likely be joined by two new members. The team will continue maintaining UpStage version 3 by performing essential bug fixes. The new members will form their own proposal and we, the continuing students will undergo our mid-project review.

**Approach**

The entire semester will be planned out more thoroughly nearer to the time. The approach that is chosen to use for this part of the project will be reviewed to see how effective it was and if it needs to be changed.

**Priority**

The main priority in this semester is to bring the new members up to speed with what has happened in the previous semester, and to get them involved with researching development of the new UpStage.

**Deliverables**

Semester 2 will likely see iterations of prototypes being developed and then reviewed to either be extended or re-designed. The three of us will also give our poster presentation.

Team collaboration and Communication

**Team Norms**

* Team members must attend all meetings. If a team member will not be able to attend the meeting, other team members will need to be notified at least one hour before the commencement of the meeting.
* If in need of assistance, ask for help from the other team members.
* There must be mutual respect between the team members.
* The team must act as one not two. There must be a collaborative environment and open dialogue between the new members and the old team.

**Communication Plan**

|  |  |  |
| --- | --- | --- |
| **Who** | **What** | **How** |
| **Project Supervisor**  Anne Philpott | Project Progress  Project-related Issues  Project Deliverables | Face-to-Face meetings  Email |
| **Project Team**  Vanessa Henderson  Nikos Phillips  James Williams  Takuma Sato  Xiangyu Chen | Project Progress  Project-related Issues  Project Deliverables | SCRUM Meetings  Web-based Chat  Email  Text Messages |
| **UpStage Clients**  Helen Jamieson  Vicki Smith | Project Progress  Project Deliverables  Project Requirements | Email via Team Gmail account  Face-to-face meetings  Online meetings |

Glossary

*Acceptance test -* A process to verify if the work has satisfied the requirements specified in the user requirements document.

*Avatar* - An icon to represent a particular person on the stage.

*GitHub* - Web-based hosting service of software development project.

*Regression test -* A process of verifying added changes to software haven’t affected the rest of the program and it is still functioning correctly.

*SCRUM* - an iterative and incremental [Agile software development](http://en.wikipedia.org/wiki/Agile_software_development) framework for managing software projects and product or application development (Wikipedia)

*Sprint*  - A short period of time where work is assigned to be completed

*Story point* - a value assigned to a task to be completed. A ratio is assigned to determine how many hours a story point is worth. This can be 1:1

*SVN* - Abbreviation of “Apache Subversion” which maintain current and historical version of files.

*Text-to-speech* - Artificial sound of human speech created by computer system. Also known as speech synthesizer.

*User story -* A simple way of capturing the user requirements. These should be as concise as possible and from the user’s point of view.

References

1. Bipp, T., et al. (2008). "Pair programming in software development teams – An empirical study of its benefits." Information and Software Technology 50(3): 231-240.

2. Ilieva, S., Ivanov, P., & Stefanova, E. (2004). Analyses of an Agile Methodology Implementation. doi:10.1109/EURMIC.2004.1333387

3. Wikipedia, *Scrum (Software Development)* retrieved March 24, 2014 from http://en.wikipedia.org/wiki/Scrum\_(software\_development)

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Modified By |
| 13/03/2014 | 1 | Created main structure of document with headings for important sections and bullet points | James, Xiangyu |
| 16/03/2014 | 2 | Added vague details in each section for confirmation with Anne on Monday 16/03 | Takuma, James |
| 17/03/2014 | 3 | Added more details to Quality Assurance entry | James, Xiangyu |
| 20/03/14 | 4 | Added schedule details and modified text into subheadings with relevant content | Takuma, James |
| 25/03/14 | 5 | Edited entire document according to Anne’s guidance | Takuma, James, Xiangyu |

**Auckland University of Technology**

**Bachelor of Computer & Information Sciences**

**Research & Development Project**

**Disclaimer:**

**Clients should note the general basis upon which the Auckland University of Technology undertakes its student projects on behalf of external sponsors:**

*While all due care and diligence will be expected to be taken by the students, (acting in software development, research or other IT professional capacities), and the Auckland University of Technology, and student efforts will be supervised by experienced AUT lecturers, it must be recognised that these projects are undertaken in the course of student instruction. There is therefore no guarantee that students will succeed in their efforts.*

*This inherently means that the client assumes a degree of risk. This is part of an arrangement, which is intended to be of mutual benefit. On completion of the project it is hoped that the client will receive a professionally documented and soundly constructed working software application, some part thereof, or other appropriate set of IT artefacts, while the students are exposed to live external environments and problems, in a realistic project and customer context.*

*In consequence of the above, the students, acting in their assigned professional capacities and the Auckland University of Technology, disclaim responsibility and offer no warranty in respect of the “technology solution” or services delivered, (e.g. a “software application” and its associated documentation),both in relation to their use and results from their use.*