**Project Plan**

***Client***

**Students**

***Project Members***

**Jeremy Dubansky**

**Chris Tedford**

**Aaron Peterson**

**David Vriezen**

**Porous Porcupines**

**Table of Contents**

Problem  
Feasibility  
Risks  
Goals, Questions, Metrics  
Economics  
Prototype   
Requirements   
Design  
Module interface specifications  
Initial module implementations, for minimal subset  
Module tests, including test results for implemented modules  
Module implementations   
V&V plan  
WBS  
Summary of Manageable Risks  
Review results

Retrospective report, including measures

# Problem(s) to be solved

Students are looking for a good degree map that is as fluid as their personal schedules are. The problem arises when a student might decide to change majors yet would like to see what the options are with the classes they have already completed. This is also helpful for students who might be just entering school as well because it shows them a path to graduation. Transferring students might also find this appealing because they can add in the classes they have already completed somewhere else and view the options here. There are current degree maps but they lack the ability to adapt to possible other majors. The main function of this application is to be versatile for any student to find out where they stand within their degree currently.

# Feasibility

Each component of our degree mapping application has been implemented by other developers at some time or another. Because of this, the degree of innovation is fairly low. Couple that with our team’s aggregate skill level and experience with data parsing and web development, the risk of being unable to create this application is virtually non-existent.

The largest risk to feasibility is resource cost. As the project scales up, the possible need to purchase more servers and additional man hours dedicated to maintenance will increase which may make the project uneconomical. However, our team has done a substantial amount of market research and we are selling our product intelligently and at a profitable price tag.

# Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Probability of Occurrence | Criticality (0-100) | Risk factor (Prob. of Occurrence x Criticality) | Mitigation Strategy |
| We are unable to receive necessary funding. | 0.10 | 90 | 0.10 x 90 = 9 | Our team has done research into the market and have put together a solid economic model showing that our project is a smart investment. |
| Students will not use our product. | 0.10 | 80 | 0.10 x 80 = 8 | Our architect will do a lot of student sampling and market research to make sure that our product is useful and desired |
| The course planner cannot be completed in time. | 0.05 | 40 | 0.05 x 40 = 2 | We have budgeted a generous amount of time to complete this project, and have budgeted extra funds for incidental costs |

Risk 4:

The team will be unable to develop the necessary software.

0.01 x 60 = 0.6

Mitigation: Our team has already done quite a bit of functional prototyping proving that we are able to create the core features of our product.

Risk 5:

Our product fails to sell to universities and becomes uneconomical.

0.30 x 80 = 24

Mitigation: Our team is doing a substantial amount of research into the market and we are creating a product that will be desirable to students and advisers.

# GQM

|  |  |  |
| --- | --- | --- |
| Goal | Question(s) | Metric(s) |
| Make this product accessible to as many university students as possible with no cost to the users. | How much will it cost to develop the system? | What are the wage costs?  How long will development take? |
|  | How much will it cost to maintain the system? | Software maintenance costs as a function of time. |
|  | How much will it cost to scale the system up?  How much will we have to charge to keep costs away from users? | Server maintenance costs as a function of time.  Costs to add a new server.  What average bandwidth do we need to record before adding a server?  Make back costs on university-wide licenses and setup fees. |
| Produce a product capable of competing with the current university planning tools. | How useful will students find our product? | Quantitative survey results from test groups. |
|  | What is the necessary user coverage we need to determine whether we are meeting our audience’s wants or not. | Percentage of students we have received feedback from. |

# Economics

Our team plans to create scheduling software, which universities will buy and provide to their students hosted on their own servers. We will charge them a one-time licensing and setup fee of $5000. With these numbers, we will need to sell only four units to break even.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Quantity | Unit Price | Total |
| **Human Costs** |  |  |  |
| 4 employees | 320 manhours | $30/hr | $9,600 |
| **Installation Costs** |  |  |  |
| 1 Employee | 30 manhours | $30/hr | $900 |
| **Technical Costs** |  |  |  |
| Development Server | 1 month | $50/mo | $50 |
| Office Space | 1 month | $900/mo | $900 |
| **Misc** |  |  |  |
| Emergency Fund |  |  | $3000 |
| Total Cost to First Unit |  |  | $14,450 |

# Prototype

The initial prototype for this application will take the form of a user interface mock-up with basic scheduling functionality for a single degree path. By releasing this prototype to our client for evaluation and updating the interface based on user feedback, this can be used as a way to implement our client’s needs with an optimal interface in the final product.

Further prototyping can be accomplished during an interim verification and validation phase, wherein the product prototype will use the path-mapping algorithm and other essential functionality achieved to the date of verification, both implemented with respect to a single degree path. This will produce a fundamental iteration of the final product, which may then be scaled up and augmented with further components both to an initial version for wider use within Iowa State and to potential later iterations for use with other institutions.

# Requirements, including uses cases

**Functional Requirements:**

* Users must be able to log in and log out.
* Users must be able to delete their accounts.
* Users must be able to update and reset their password.
* Users must be able to select which courses they have already taken.
* Users must be able to select which majors and minors they want a course plan created for.
* Users must be able to view the computed best schedule for them.
* Users must be able to view other computed non-best schedules.
* Administrators must be able to add, remove, and update the courses offered.

**Non-functional requirements:**

* Access to user accounts must be secured.
* Access to the course database must be secured.
* The software must be able to run with without crashing.
* The software must be able to calculate a schedule quickly.

# Design

Given the fact that our primary user base consists of college students, the design is going to be a very important aspect to who uses the product. It needs to be simple enough to use, work effectively, and also have a clean, friendly appearance. The design can be created using Graphical User Interface (GUI) libraries from JavaScript. There is a very public library known as jQuery that include many valuable tools for creating a good front end.

# -modular structure

The modular backend is mostly developed in JavaScript and jQuery to help push data in the correct places. Two major modules being played by JavaScript is the XML parsing as well as taking care of all the GUI aspects. Dealing with XML is relatively easy with Javascript, it allows us to access the contents basically in the form of a tree. From this data we can display it in any form as JavaScript also has a large library of GUI components.

# Module interface specifications

The essential modules this project needs is a clean GUI and a good storage system for the data. When thinking of system that combined both these aspects we looked JavaScript web development. Many local applications lack the cleanliness a web page contains and may require installing other packages. This project’s modules are simple skeletons for reading and writing to XML and the ability to have a fluid GUI. With both interfaces being very simplistic we can add to them what we need, changing files or how output is dealt with can be customized to the needs of the smaller modules.

# Initial module implementations, for minimal subset

The initial implementation as used in the prototype for the schedule planning tool uses an XML file to store data locally, processes this information using JavaScript and display it using jQuery. The modules created for the few functions necessary are simple interface designed to do multiple tasks. The idea behind this concept is effective because testing on one module such as reading from a certain file, can be tested anywhere. Once a module is working and has been tested it can be dropped into place and customized to do the job for that area.

For more information, refer to the pointer to our repository.

# Module Tests

With html and PHP being so versatile we are easily able to test areas of our project in modules. Being just a blank document that simply deals with output to the screen we have a great debugging tool. When working with something like JavaScript and PHP it is easy to view output at almost any given point in the application. Javascript works in a very modular way in which there are classes but classes are not needed to run code against. If a section of JavaScript is broken we can simply remove it to another location and fix it and move the edited code back in. This allows us to continue working on the rest of the code without the big picture not working due to a small section.

Many tests were ran against certain areas of code individually before being placed in the larger program. It is important that the code be runnable on not just one machine but as many machines as possible without there being any issues. Due to the fact our code is web based there will always be an atmosphere where our code my not work fully. Testing is involved to ensure that most browsers and platforms can easily access and use all the features of the project.

# Module implementations

With this project being web based we found a simple way for us to collaborate. Unfortunately GIT is difficult to implement when working in a group on a web based platform. The problem arose when trying to access xml that was local and the links pointing to out-of-date data. There are a few pushes for whenever there was a large portion of the site completed, but it was better to download the current version from FTP. When any modification might be made, it is simpler to just FTP the page back up and test again an atmosphere that works.

Git: <https://github.com/dubansky/329Porous>  
Site: <http://dubansky.com/329/>

# V&V plan

We will use the steps below to validate and verify our product.

1. Reviews
   1. Requirements will be reviewed by key stakeholders
   2. User Interface mockups will be reviewed by key stakeholders and potential customers.
   3. Application prototypes will be reviewed by key stakeholders and potential customers.
   4. Potential customers will provide beta testing feedback.
2. Code Reviews
   1. Non-critical code will be informally reviewed by at least one other programmer, before being merged into the master branch.
   2. Critical code will be formally reviewed by the entire team before being merged into the master branch.
3. Testing
   1. We will use testing tools to simulate stress testing for thousands of users.
   2. We will use white-box testing and black-box testing to ensure that the program outputs correct “best schedules” for given schedule configurations.
   3. Regression testing will be used to ensure that changes to code do not cause tests to fail.

The acceptance criteria for the first and subsequent releases are as follows:

1. Zero critical defects
2. Zero major defects
3. A maximum of 10 minor defects
4. Zero errors in the specifications of any black-box test cases.
5. Code coverage in testing of 60%
6. Branch coverage in testing of 70%
7. The product, installed on a properly configured server, can handle 2500 users without negatively impacting the product’s performance.
8. The produce will produce the “best schedule” for every valid input.
9. The product, under normal load, will produce a “best schedule” in under 2 seconds.

# WBSC:\Users\envy\Desktop\329\WBSPorous.png

# Summary of Manageable Risks

There are a few road blocks in a project of this size, but these can fixed with a good outlook at the next stage in development. One of the larger risks that is on most every project is, will there be a market for this when it is finished, and there is only one way to find out. To reduce that large risk to a smaller version the team has to eliminate other risks and product the best product in the market. Throughout production there are multiple coding risks involved, nothing that cannot be fixed with a closer look at the modular design.

# Review results

Although no formal reviews have taken place thus far, tests have provided a positive outlook on the project. With a fully functional application we can see more positive reviews and further the project based on these reviews. The product can be a simplistic answer to the problem at hand and through reviews and tests turn out to be a much better product. As it stands the reviews are the building blocks of what the project could eventually turn to be. The intention is it use reviews of the project and build what we need on the base of the application.

# Retrospective report, including measures

To ensure we wrote the correct application it is important that the users are getting the best experience possible. The user experience is a key factor in keeping members to use this solution over others. Using data from page views and simple questionnaires we can determine if the project is working. It should be noted that this application will not see use every day by every user, the intention is only around scheduling class times. Around these periods of higher usage we could have a poll on new features and ask how users enjoy the product at its current state.  
  
A very good measure to rate this product is the amount of usage the stakeholders working on the project use it. The project itself is helpful to those who work on the actual application, it should be everything they intend it to be as well. The product can always be fined tuned to what the customer wants, but it is helpful if the people making the product are also the ones using the product.