Product/Report Title Here

Senior Design Final Documentation

DanceSoft

Marcus Berger Dicheng Wu

December 8, 2015

Contents

List of Figures

List of Tables

List of Algorithms

Overview Statements

0.1 Mission Statement

The mission of the DanceSoft team is to create a efficient and effective system for the Academy of Dance Arts. So data can be managed clearly and easily to manage the academy's day to day needs. [MB]

0.2 Elevator Pitch

The DanceSoft project is a data management software for the Academy of Dance Arts in Rapid City. The project aims to produce a simple and more effective data management tool then what is currently in use at the academy. This will be accomplished through the use of a database and a simple user friendly interface, which will allow faculty and students to easily accomplish their needs, whether that's registering for a class, getting a role sheet or just general people management. DanceSoft will provide effective management tools so people spend less time at their computer, and more time dancing. [MB]

Document Preparation and Updates

Current Version [X.X.X]

Prepared By:
Marcus Berger #1
Dicheng Wu #2

Revision History

Date	***Author***	***Version***	***Comments***
9/12/15	*Marcus Berger and Dicheng Wu*	*1.0.0*	*Initial version*
10/8/15	*Marcus Berger*	*1.0.1*	*Sprint Report 1*
10/22/15	*Marcus Berger*	*1.0.1*	*Initial Overview*
10/23/15	*Marcus Berger*	*1.0.1*	*Initial Mission Statement and project.tex*
10/28/15	*Marcus Berger*	*1.0.1*	*Initial Requirements, Testing, and develop.tex*
11/5/15	*Marcus Berger and Dicheng Wu*	*1.0.2*	*Sprint Report 2*
11/5/15	*Marcus Berger*	*1.0.2*	*Updated Overview*
12/4/15	*Marcus Berger and Dicheng Wu*	*1.0.3*	*Sprint Report 3*
12/8/15	*Marcus Berger*	*1.0.3*	*Updated Project, Uploaded Resume, and Add Acknowledgments*

1

Overview and concept of operations

This chapter contains a general overview of the DanceSoft project. [MB]

1.1 Scope

The scope of this project is to develop a system that allows the academy to manage it's activities and information in an effective manner. Allowing the academy to manage their employees and their students.[MB]

1.2 Purpose

The purpose of this document is to describe and detail the major components of the system and how they will be developed. The major components of the application are as follows:

1. Major System Components
2. System Goals
3. Concept of Operations (CONOPS)
4. System Overview
5. Technologies Used

[MB]

1.3 Deliverables

Listed below are the deliverable major system componets for this project. [MB]

1.3.1 Major System Component: Database

The first major component of the system is the MySQL relational database. The database contains the Academy's data and is the core of the back-end side of the software. This database will be the conduit for most of the systems interactions with the data. The database will live within a local computer provided by the user.

1.3.2 Major System Component: User Interface

The second major component of the system is the front-end user interface for admins and teachers. This will be the only part of the system most users ever see, and will provide an effective means to complete the desired user task. This is accomplished through the use of pages created in PyQt with interfaces to give users effective ways to interact with the database and the necessary data for their requested operations.

1.3.3 Major System Component: Student Interface

ADD LATER (AFTER WINTER BREAK)

1.4 System Access

Listed below are the various access information, or credentials for the various levels of the system. [MB]

1.4.1 Local System GUI

1.4.2 Student Web Interface

1.4.3 Database Access

ADD LATER (AFTER ACCESS IS FINALIZED IN DEPLOYMENT BOX)

1.5 Systems Goals

The system needs to provide a solution which can run the dance studio data and some day to day activities in an effective and secure manner. This includes allowing teachers to print role sheets, look at schedules, and manage their information. Students need to have the ability to see information pertinent to them such as registration and class requirements. Owners need to be able to use the system to manage their employees, the academy's students and it classes, and other administrative duties such as billing and payroll Lastly this system as a whole also needs to be an improvement on the current system in use by the customer and provide an easier and more efficient way to run the clients business.

Overall the system goal is to provide a environment where academy owners, teachers, and students can effectively manage their personal needs and requirements for academy participation and continued operations.

1.6 Concept of Operations

This section will cover how the system is used from the viewpoint of the various types of users.

1.6.1 New Users

1.6.1.a New User: Admin

1.6.1.b New User: Teacher

UPDATE AS NEW USER PROCESS IS LAID OUT (MOST LIKELY END OF SPRINT 4)

1.6.1.c New User: Student

UPDATE AS STUDENT INTERFACE IS DESIGNED

1.6.2 Existing Users

This section will cover the options for a user already established in the system.

1.6.2.a Existing User: Admin

Landing Page This page contains the Admin related options

Manage Employees This page contains the list of admin functionality associated with the employee functionality

Search Teachers The user is brought to a new window where the user can enter a employees name and see a list of matching teachers. This can also be done using advanced search to search on other fields such as phone number or address. After the search is completed then the user can click on a result to bring up more detailed information and change it as needed.

Update Teacher The user is prompted with a form where they can select a teacher. After teacher selection the form is populated with that teachers information, the user can then change the information as needed and click the submit button to confirm the changes and submit to the database.

Assign Teacher To Class Opens a window that contains the list of classes ordered alphabetically. The user then selects a class, after class selection the pages generates a list of available teachers by cross referencing their current classes. A teacher is then selected and confirmed before submission to the database. If the selected class is already assigned then a dialog ask the user to confirm the teacher change if yes then the user selects a new teacher and confirms the selection.

New Teacher The user is prompted with an empty form to input teacher information. The user clicks submit when finished and confirms the information before submission to the database.

Manage Classes This opens a page containing the list of admin functionality relating to class management.

View Class UPDATE WHEN FUNCTION FINALIZED

Modify a Class The user is prompted with a form where they can select a class. After class selection the form is populated with that classes information, the user can then change the information as needed and click the submit button to confirm the changes and submit to the database.

Add a Class The user is prompted with a form, and asked to fill out the necessary information and click the submit button. This then produces a dialog box to confirm the submission and make the final add to the database.

Manage Students This page contains the list of admin functionality associated with the student functionality

Search Students Like the employee search above, the user is brought to a new window where the user can enter a student's name and see a list of matching students. This can also be done using advanced search to search on other fields such as phone number or address. After the search is completed then the user can click on a result to bring up more detailed information and change it as needed.

Modify Student Information The user is prompted to enter a student name and if the student exist the form is populated and the user can change the information as needed and updates can be submitted to the database. (FUNCTIONALITY CURRENTLY BEING REEVALUATED MAY MERGE WITH SEARCH FUNCTION)

Add/Remove Student From Class The user is given a list of classes, the user then selects a class. After the class is selected the students currently in the class are displayed. The user can then select a student and click the remove button, or they can click the add button to add a student that has a pending registration to the selected class.
(need user story updates to see if only pending students should show up)

Registration **UPDATE AS FUNCTIONALITY IS ADDED**

1.6.2.b Existing User: Teacher

Student Options **This buttons opens up the the teachers student related functionality.**

Modify Student Info When the user selects this option the system will open up a form where a student can be selected and the form will be populated. Then the user can modify the information in the form and submit the updates to the database

Search Students The user is brought to a new window where the user can enter a student's name and see a list of matching students. This can also be done using advanced search to search on other fields such as phone number or address. After the search is completed then the user can click on a result to bring up more detailed information and change it as needed.

Add Student to Class Much like the Admin function for adding and removing students the teacher will get a window where they can select one of the classes they teach. After that the teacher can select a pending student and add them to the class. (FUNCTIONALITY MAY CHANGE WITH USER STORY CLARIFICATION)

See Schedule The user will open a window which will show them what classes a selected student is enrolled in on which day. (MORE INFORMATION AS FUNCTION IS COMPLETED)

Class Options This buttons opens up the the teachers class related functionality.

See Schedule The user will open a window which will show them what classes they are teaching on which day. (MORE INFORMATION AS FUNCTION IS COMPLETED)

Class Info The user can select a class and view the class information.

Role Sheet A window comes up where the user can select one of the classes they are currently teaching and the list of students in the class. The user can then hit a print button which will open up a secondary window where the user can modify the list, import a list from a file, preview the list as a pdf, and print the list.

Personal Options This buttons opens up the the teacher functionality relating to personal information.

Enter Hours (WORKED ON IN SPRINT 4

Modify Information (MAY DROP FUNCTIONALITY SINCE IT SHOULD ONLY BE AN ADMIN FEATURE)

RESET PASSWORD User will be prompted to enter and confirm there current password and then if the passwords match the users password then the user will be propted to enter a new password after checking password it is updated in the database.

1.6.2.c Existing User: Student

UPDATE AS STUDENT INTERFACE PROGRESSES

1.7 System Overview and Diagram

Users will access this application through a local GUI or a web application depending on their position within the system. The GUI will be located in the academy local box, and the student web application will be excess able through any web browser. When a user makes a connection to the website, the pages and data will be sent to the user. Figure 1 shows an overview of the student interface architecture. When a local GUI user connects the user will

navigate through the vareous pages listed above to the desired functionality. Figure 2 shows a simplistic view of the GUI Architecture. There are three major components to the project, database, student interface, and admin/teacher interface. Each section is described in more functional detail above and in section 4 Design and Implementation. (ADD MORE DETAIL AS SYSTEM FINALIZES) [MB]

(ADD FIGUREs AFTER SYSTEM DESIGN FINALIZATION)

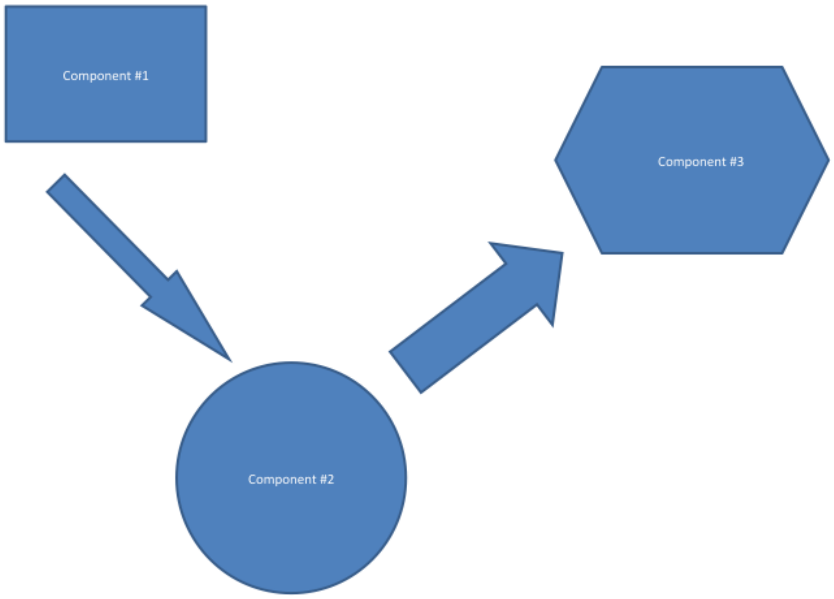


Figure 1.1: A sample figure .... System Diagram

1.8 Technologies Overview

The primary Technologies for this projects are as follows:

1. **Xcode and Visual Studio - Xcode and MSVS are the primary development IDEs for this project. Of the two the one the group used the most was visual studio so we could develop on the machines provided by the school.**
2. **Python - the primary programming language for the project**
3. **PHP - primary language for student web interface**
4. **PyQt and Qt Designer - Gui package and development environment**
5. **MySQL - MySQL provides the database and relational quires to manage the data and organize it within the system.**

These technologies were selected after a first research sprint where research into programming language, GUI, and database options were selected. A brief description of the research can be found in the sprint 1 report or the first prototype sections of this document. [MB]

2

Project Overview

This section provides some information with regards to the team roles, project management, and phase overview.[MB]

2.1 Team Members and Roles

The DanceSoft team consist of two members, Marcus Berger and Dicheng Wu.

Marcus Berger(Scrum Master/Development Team) - As a member of a two person team the roles for this project blend together significantly. Both team members mostly do equal shares of all work types. As the primary manager of the DanceSoft Trello board, Marcus has mostly taken on the role of scrum master within the group. However his primary role is still development.

Dicheng Wu(Development Team) - Dicheng's primary role as with both members of the team, is development of the software. However like the other members since the team is so small each member of the two person team must be able to fill all needed roles within the team.

Dr. Jeff Mcgough(Product Owner) - While not a working member of the team Dr. Mcgough is a secondary scrum master and product owner to the group due the small size of the team. Main duties in this role include talking to the client about what is needed, and making sure the team stays on task and is going in the correct direction based on the clients needs

Author notation will be identified by [MB] for Marcus and [DW] for Dicheng at the start of major sections (ex: after heading 2.0 above). Subsections within the main sections assumed to be by the same author. [MB]

2.2 Project Management Approach

This project is managed using several service and an Agile sprint based methodology. The sprints which range in purpose from research to development are three weeks long with a week after each sprint for review and possible presentations to the client. The project is broken into several users stories, which are then broken down into a project backlog of task for development. The stories and backlog are discussed in a later section of this document. Project management tools used by the team are Github, for source control and code management, and Trello for sprint and backlog management. Trello provide a list of task to be accomplished in each sprint and the ability to see modified tasks and keep track of team members current assignments. The task assign in each sprint are decided be the team with some input from Dr. McGough as a secondary scrum master. These assignment can be altered or changed based on product owner input. Sprint reports are sent to Dr. McGough for both project owner and grading purposes at the end of each sprint. Scheduled Meeting occur every Tuesday and Thursday from 10:00-12:00, other times are assigned as the team finds necessary. The team also has a weekly meeting with the product owner Dr. Jeff McGough every Wednesday at

2:00 where the progress of the project, the next steps, and any lingering questions about the project can be addressed. [MB]

2.3 Phase Overview

This project will be implemented in phases, the phases follow. [MB]

2.3.1 Initial User Story and Requirements Gathering

This phase consist of meaning and discussing the user stories and requirements with the product owner. The product owner also lays out limitations and constraints for the project. More information can be found in section 3.0 Requirements

2.3.2 Database Creation and Starting Pages

During this phase the database schema will be constructed and implemented, being sure to keep the schema as concise as possible. The goal is to generate a database that can effective support the needed functions and GUI connectivity. After the database is constructed, the team will move on to the starting landing pages which will be jumping off pages for functionality creation. These page will almost certainly be modified as phases progress.

2.3.3 Prototyping and Testing

This phase will conpose the bulk of the project as the first development of the functionality requested by the project owner are constructed. As the prototype progresses check ins with the client will be conducted to be as sure as possible that the team stays on track. Pages will be tested as the pages are constructed.

2.3.4 Development Updates and Testing

This phase will consist of updates to existing functions, updates, and prototyping updated, new, or misunderstood functions. Testing will be conducted as updates are made to ensure new pages work, and updated pages retain requested function. During this phase user stories and requirements should move to final completion.

2.3.5 Final Testing, Production, and Delivery

During this phase the last bit of testing on the remaining functionality will be conducted. After this the final production needs will be completed, this may consist of final deployment or other steps depending on the state of the project at this time. Lastly the project will be handed over to the client for delivery and the senior design fair. By this point the product itself will live on the box provided by the client and a production server. Access Information, logs, and user guides and manuals will be provided, with the user guide as both pdf files and psychical copies.

2.4 Terminology and Acronyms

1. GUI - Graphical User Interface - the front end screen that the users interact with using graphical assets

(ADD TO SECTION AS NEEDED

3

User Stories, Backlog and Requirements

3.1 Overview

The overview should take the form of an executive summary. Give the reader a feel for the purpose of the document, what is contained in the document, and an idea of the purpose for the system or product.

The user stories are provided by the stakeholders. You will create the backlogs and the requirements, and document here. This chapter should contain details about each of the requirements and how the requirements are or will be satisfied in the design and implementation of the system.

3.1.1 Scope

This section covers the purpose of systems, the client's information and restrictions on the system, and the user stories for requirements.

3.1.2 Purpose of the System

The purpose of this system is to provide a system for the Academy of Dance Arts to manage their day to day operations, their employees, and their s friendly students. These day to day operations range from assigning teachers to classes, looking up information, printing roles sheets for classes, etc. Also the system must accomplish these task using a user friendly interface, that is as intuitive as possible. Students will also have a simple web interface to look up their information and register for new classes.

3.2 Stakeholder Information

The stakeholder and sponsoring customer for this project is Dr. Jeff McGough a computer science professor at the South Dakota School of Mines and Technology and the vice president of the Academy of Dance Arts in Rapid City, South Dakota. Well not the sponsor of the project Dr. McGough's wife, Julie McFarland is also a key part of the customer base and a stakeholder as the owner of the Academy.

3.2.1 Customer or End User (Product Owner)

The primary end user for this product is Julie McFarland and her employees to manage the Academy of Dance Arts. Julie well not playing a direct role in product development is able to convey the academy's needs through Dr. McGough.

Dr. Jeff McGough is the primary point of contact in the project. He assumes the role of scrum master at times and drives the product backlog and provides more details on product backlog materials during the weekly meeting with the development team.

3.2.2 Management or Instructor (Scrum Master)

Dr. Jeff McGough is the primary point of contact in the project. He assumes the role of scrum master at times and drives the product backlog and provides more details on product backlog materials during the weekly meeting with the development team.

3.2.3 Developers –Testers

The DanceSoft team consist of two members, Marcus Berger and Dicheng Wu, who are both primarily developers and testers. due to the fact that the team is so small. All development roles are shared between the two team members.

3.3 Business Need

The customer needs us to develop a software solution which can run the dance studio in an effective manner. The product also needs to handle changing classes from year to year without needing to be updated. This means that the software needs to sync with multiple users, and handle new information such as class rosters, prices, clothing requirements for classes, changes in the employment roster, and many other changes that can occur in the running of a dance school.
This project as a whole needs to be an improvement on the current system in use by the customer and provide an easy and efficient way to run the clients business. This project will the data manipulation task through the back-end MySQL database, and the ease of use will be handled with a simple Qt Gui

3.4 Requirements and Design Constraints

This section discusses what requirements exist that deal with meeting the business needs the customer has. For the DanceSoft these include system needs to run the academy, network connection issues, and some environment requirements laid out by both the client and the senior design requirements.

3.4.1 System Requirements

The system requirements laid out by the clients are just the necessary features laid out in the user stories below. There was no preference on language or GUI environment on the part of the client. Due to some of the user stories and information handled within the system a level of security becomes a system requirement.

3.4.2 Network Requirements

The network inside the Academy has connection issues and therefore a cloud or online based data storage option is not a possibility. The network issues within the school mean the system will be contained in a local system to provide more stability.

3.4.3 Development Environment Requirements

The academy runs on a Mac so the system must work on the mac OSX operating system. While not required the project is developed in Python, so the end product will work cross platform should the academy ever switch operating systems.

3.4.4 Project Management Methodology

The client requires a weekly meeting every Wednesday at 1:00 p.m. to check on the progress of the system. These meeting vary on topic and length depending on the needs of the project and the status of task. The senior design class requires that this project be completed in six sprint that are all roughly three weeks long, with a week long results period after each one. Another project requirement is the presentations that are required by the senior design class. These presentation occurs twice every semester usually after the first and last sprint each semester. Each presentation cover the content of the project up to that point, and updates on topics such as risk mitigation, budget, and current prototypes. Lastly it was requested by Dr. Mcgough as part of senior design and as the client that we provide him with access to the Github repository for the project and the Trello board.

3.5 User Stories

After the requirement for the project were laid out the team created the user stories based on those requirement. The user stories the team came up with are as fallows:

1. As a user i want to adjust students payment models
2. As the owner I would like to see automatic database backups.
3. As a student I would like to be able to register online (with special app). Classes must be approved before added.
4. As a student I would like to be able to search clothing requirements.
5. As a student I would like to know my billing.
6. As the owner I would like to indicate clothing requirements per class.
7. As a studio person, I would like to be able to add students to classes.
8. As a student, teacher etc, I would like to be able to look up a students class list.
9. As the teacher I would like to get a class role for each class.
10. Given a class list, I would like to get an invoice of the tuition due.
11. Studio would like to track payments and estimate remainder due. I would like to generate an invoice for this amount.
12. As a student I would like to be able to register online (with special app). Classes must be approved before added.
13. As a student I would like to know my billing.
14. As the owner I would like to track teacher hours and compute payroll.
15. As a studio employee I would like to open a registration pane and add student data
16. As the studio owner I would like to enter teacher information and look up information such as SS and pay rates.
17. As the owner, I would like to enter classes: time, location, registration cap. I would like to view this information later. I would like to assign instructors
18. As a user I want to have different payment models for different situations

3.5.1 User Story #1

As a user i want to adjust students payment models. This user stories means that the user should be able to go find a student, select that student and change the pa model to another existing payment model. Adding a payment model is part of a different user story.

3.5.1.a User Story #1 Breakdown

Further breakdown for this user story could be the creation of the student select function but this is mostly covered by other user stories.

3.5.2 User Story #2

As the owner I would like to see automatic database backups. This one is fairly simple the system will need to back up the data from the database locally or by an external provider.

3.5.3 User Story #3

As a student I would like to be able to register online (with special app). Classes must be approved before added. The special app references in this user story is the php student web interface that the team will create. The students will be able to go to the website, log in and register for classes along with other features listed in other user stories.

3.5.3.a User Story #3 Breakdown

Also listed in this user story is the ability for admins to approve any class registrations by students before the registration is finalizes and submitted to the data base. This approval system needs to have three states. First is pending which will be the unanswered request in the system. Second is the approved option which will finalize the students registration and place them in their desired class. Lastly is denied which will not put the student in the class.

3.5.4 User Story #4

As a student I would like to be able to search clothing requirements. This will be part of the student interface and will allow them to click on a class and see clothing requirements for that class from the database.

3.5.5 User Story #5

As a student I would like to know my billing. The students will be able to click a link within the interface and see their various billing info, such as remaining balance, payment plan, and date payment is due.

3.5.6 User Story #6

As the owner I would like to indicate clothing requirements per class. The owner or other admin will be able to add clothing requirements to a specific class and change them in a class menu.

3.5.7 User Story #7

As a studio person, I would like to be able to add students to classes. This options will allow all employees to request a specific student be added to a class. This will sent a request to an admin which will need to approve the request like a normal registration.

3.5.8 User Story #8

As a student, teacher etc, I would like to be able to look up a students class list. Users need to select a student and see what their class schedule is and which class they have pending registrations for.

3.5.9 User Story #9

As the teacher I would like to get a class role for each class. Users need to select a class and see who is enrolled in it. Also the list needs to be exportable or printable so teeachers can take role at a class.

3.5.10 User Story #10

Given a class list, I would like to get an invoice of the tuition due. Users should be able to get an invoice for their billing based on the number of classes being taken, and the payment model the student is placed under.

3.5.10.a User Story #10 Breakdown

This user story along with any others that deal with payroll or billing with need to have a system for amount calculation and a way to adjust or add payment models on the admin/owner side of the software.

3.5.11 User Story #11

Studio would like to track payments and estimate remainder due. I would like to generate an invoice for this amount. This user story fallows user story 10. The system should track payments made and given those payments calculate what students or parents have left to pay.

3.5.12 User Story #12

As a student I would like to know my billing. Simply put students and parents want the abilite to see how they are charged and what they have left to pay. This should be an option within the php based student web interface.

3.5.13 User Story #13

As the owner I would like to track teacher hours and compute payroll. Hours for teachers will need to be approved by an admin within their interface. Also calculations will be made based on that teachers pay rate to compute their pay. Lastly tax algorithms will need to be used to effective make sure that tax are withdrawn correctly and neither the teacher or the academy will be liable in the case of audits.

3.5.13.a User Story #13 Breakdown

Teachers and employees will need to be given a way to submit hours through their interface that can then be approved by the owner.

3.5.14 User Story #14

As a studio employee I would like to open a registration pane and add student data. The employees of the academy should be able to modify student registration information. This will be used should the students information change, or if the students information was entered incorrectly.

3.5.15 User Story #15

As the studio owner I would like to enter teacher information and look up information such as SS and pay rates. The system will provide the owner with the ability to search, view, and modify information within the system.

3.5.15.a User Story #15 Breakdown

Search and view functions will exist for all level of users with different results in different areas. Examples being students need to see class information, teachers should search classes and students, admin should be able to see all information.

3.5.16 User Story #16

As the owner, I would like to enter classes: time, location, registration cap. I would like to view this information later. I would like to assign instructors. Simply put this is the ability of the owner to a class to the academy's roster.

3.5.17 User Story #17

As a user I want to have different payment models for different situations. Allow the owner the ability to change, add, and select different payment methods for billing based on a number of factors. These factors include time of registration, dropped classes, admin selects payment options for a specific situation, etc.

3.6 Research or Proof of Concept Results

Before production could begin research had to be conducted into which programming language, GUI framework, and database type would be used to complete the project. A explanation of the research conducted can be found in the sprint one wrapper or in the prototype sections of this document. After this research was conducted the team selected Python, PyQt, and MySQL as the language, GUI, and database respectfully.
After the research no explicit proof of concept was required.

4

Design and Implementation

This section is used to describe the design details for each of the major components in the system. Note that this chapter is critical for all tracks. Research tracks would do experimental design here where other tracks would include the engineering design aspects. This section is not brief and requires the necessary detail that can be used by the reader to truly understand the architecture and implementation details without having to dig into the code. Sample algorithm: Algorithm 1. This algorithm environment is automatically placed - meaning it floats. You don't have to worry about placement or numbering.

Algorithm 1 Calculate $y = x^n$

```
Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \Leftarrow 1$
  if $n < 0$ then
    $X \Leftarrow 1/x$
    $N \Leftarrow -n$
  else
    $X \Leftarrow x$
    $N \Leftarrow n$
  end if
  while $N \neq 0$ do
    if $N$ is even then
      $X \Leftarrow X \times X$
      $N \Leftarrow N/2$
    else {$N$ is odd}
      $y \Leftarrow y \times X$
      $N \Leftarrow N - 1$
    end if
  end while
```

Citations look like [2, 1, 3] and [6, 4, 5]. These are done automatically. Just fill in the database `designrefs.bib` **using the same field structure as the other entries. Then pdflatex the document, bibtex the document and pdflatex twice again. The first pdflatex creates requests for bibliography entries. The bibtex extracts and formats the requested entries. The next pdflatex puts them in order and assigns labels. The final pdflatex replaces references in the text with the assigned labels. The bibliography is automatically constructed.**

4.1 Major Component #1

4.1.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

4.1.2 Component Overview

This section can take the form of a list of features.

4.1.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

4.1.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

4.1.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

4.1.6 Design Details

This is where the details are presented and may contain subsections. Here is an example code listing:

```c
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
```

This code listing is not floating or automatically numbered. If you want auto-numbering, but it in the algorithm environment (not algorithmic however) shown above.

4.2 Major Component #2

4.2.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

4.2.2 Component Overview

This section can take the form of a list of features.

4.2.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

4.2.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

4.2.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

4.2.6 Design Details

This is where the details are presented and may contain subsections.

4.3 Major Component #3

4.3.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

4.3.2 Component Overview

This section can take the form of a list of features.

4.3.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

4.3.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

4.3.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

4.3.6 Design Details

This is where the details are presented and may contain subsections.

5

System and Unit Testing

This section describes the approach taken with regard to system and unit testing.

5.1 Overview

The testing for this project will be done using a unit test approach usually done through provided frameworks provided inside either Visual Studio or Xcode. Also user test will need to be conducted on the user interface since code based unit test can not be written for certain parts of the user interface, such as navigation flow. The test will be small pieces of codes ment to test small sections of code for success or points of failure. The overall goal will be to get around 80 percent code coverage and to test database response and responsiveness.

Provides a brief overview of the testing approach, testing frameworks, and general how testing is/will be done to provide a measure of success for the system.

5.2 Dependencies

Currently the project contains a few known dependencies in a testing environment. The first is in the PyQt4 Python library which if changed could cause issues within the system GUI and functionality test written for it. However this seem unlikely since the main focus of PyQt updates is now focused on PyQt5. Second is the project dependence on MySQL relational database. This dependency should also be negligible since any update to MySQL are normally done with continuing compatibly with existing software in mind.

Describe the basic dependencies which should include unit testing frameworks and reference material.

5.3 Test Setup and Execution

Test cases for this project were develop by the team to test specific functions and code sections and are therefore developed by the coder to test sections of the code. This can range from a print to test loops or function access to produce a full data entry or a response to test database connects and data modifications. Depending on the type of test needed the development can take many forms. Such as specific script file test, in which case the test will execute every time the script is run until the test is removed from the code of the script. This form of test is usually used by the team to test loop or other condition statement functionality or any other one off type of code test. Second is GUI test, user interface test will be conducted in two ways. Buttons and pathways will be manually tested, this means running the program and clicking the buttons to make sure that the window function properly. Second is the GUI's ability to connect and interact with the database in the correct manner. This is done through a combination of manual and script test.

———————————————— Third is the exterior system test

WRITE LATER

Describe how test cases were developed, setup, and executed. This section can be extremely involved if a complete list of test cases was warranted for the system.

6

Development Environment

The purpose of this section is to give a developer the necessary information to setup their development environment to run, test, develop and/or update the DanceSoft software.

6.1 Development IDE and Tools

The two main IDEs were used in the this project were Microsoft Visual Studio 2015, and Xcode 6. Python IDLE would also be used on occasion for minor quick fixes so the main IDEs would not need to be loaded completely. Of these the bulk of development was conducted with Visual Studio due to the fact that the laptops provided by South Dakota School of Mine and Technology run windows as their primary operating system. Visual Studio also provided a suite of debugging and testing features that allowed the team to manage and manipulate the code effectively.
The second IDE used was Xcode which is the primary development environment for the Mac operating system. This IDE was used when ever we want to directly test Mac compatability with our code. Since Mac is the required working operating system for this project. Though due to accessibility Xcode was not the Main IDE used by the team. Should the project be futher developed in the future IDE selection should not matter due to the cross-platform development of the project.

1. Visual Studio install: `https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx`
2. Visual Studio Reference: `https://msdn.microsoft.com/en-us/library/scesz732.aspx`
3. Xcode install: `https://developer.apple.com/xcode/downloads/`
4. Xcode Reference: `https://developer.apple.com/`
5. IDLE install: comes with python 3 download urlhttps://www.python.org/downloads/
6. IDLE Reference: `https://docs.python.org/3.1/`

6.2 Source Control

The source control for this product is conducted through Github. The Github repo is broken down into a final code folder, a documentation folder, and six sprint folders which contain the materials worked on in each sprint for senior design grading purposes. Connects and management is done through the provided Github GUI, which is installed when Github is downloaded. If the GUI is insufficient the group will sometimes use the shell also provided with the install. The repo is public and can be seen by anyone by going to `https://github.com/SDSMT-CSC464-F15/dancesoft/`

6.3 Dependencies

Currently the project contains two known dependencies. The first is in the PyQt4 Python library which if changed could cause issues within the system GUI. However this seem unlikely since the main focus of PyQt updates is now focused on PyQt5. Second is the project dependence on MySQL relational database. This dependency should also be negligible since any update to MySQL are normally done with continuing compatibly with existing software in mind.

6.4 Build Environment

NOT KNOWN AT THIS TIME, WILL ADD LATER

How are the packages built? Are there build scripts?

6.5 Development Machine Setup

Setup requirements to develop on this project:

1. Download Python 3: `https://www.python.org/downloads/`
2. Download necessary IDE from the links provided above or another source
3. Download MySQL: `https://www.mysql.com/downloads/`
4. Download the PyQt4 Python library: `http://sourceforge.net/projects/pyqt/`
5. Connect to the MySQL database using credentials received from the client
6. Get the python scripts and any makefiles from the DanceSoft Github repo
7. Run and test script functionality
8. Begin development updates

7

Release – Setup – Deployment

This section should contain any specific subsection regarding specifics in releasing, setup, and/or deployment of the system.

7.1 Deployment Information and Dependencies

Are there dependencies that are not embedded into the system install?

7.2 Setup Information

How is a setup/install built?

7.3 System Versioning Information

How is the system versioned?

8

User Documentation

This section should contain the basis for any end user documentation for the system. End user documentation would cover the basic steps for setup and use of the system. It is likely that the majority of this section would be present in its own document to be delivered to the end user. However, it is recommended the original is contained and maintained in this document.

8.1 User Guide

The source for the user guide can go here. You have some options for how to handle the user docs. If you have some `newpage` commands around the guide then you can just print out those pages. If a different formatting is required, then have the source in a separate file `userguide.tex` and include that file here. That file can also be included into a driver (like the senior design template) which has the client specified formatting. Again, this is a single source approach.

8.2 Installation Guide

8.3 Programmer Manual

9

Class Index

9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

10

Class Documentation

10.1 Poly Class Reference

Public Member Functions

- **Poly ()**
- **~Poly ()**
- **int myfunction (int)**

10.1.1 Constructor & Destructor Documentation

10.1.1.a Poly::Poly ( )

My constructor

10.1.1.b Poly::~Poly ( )

My destructor

10.1.2 Member Function Documentation

10.1.2.a int Poly::myfunction (int *a*)

my own example function fancy new function

new variable

The documentation for this class was generated from the following file:

- hello.cpp

11

Business Plan

11.1 Business Model

11.2 Market and Competition

11.3 Regulatory environment

11.4 Intellectual Property and Freedom to Operate

11.5 Management Team and Advisors

11.6 Sources and Uses of Capital

11.7 Financial Statements

11.8 Metrics and Milestones

11.9 Exit Plan

12

Experimental Log

For research projects one needs to keep a log of all research/lab activities.

10/15/15 Ran modified filter on data sets 1 - 6. Results were ...

10/17/15 Changed tolerance on sensor and collected data. These ...

13

Research Results

This chapter describes the results and conclusions of your research. This would be the final report for a research project.

13.1 Result 1

13.2 Result 2

13.3 Conclusions

13.4 Further work

Bibliography

[1] R. Arkin. *Governing Lethal Behavior in Autonomous Robots*. Taylor & Francis, 2009.

[2] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[3] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[4] V. Lumelsky and A. Stepanov. Path planning strategies for point mobile automation moving amidst unknown obstacles of arbirary shape. *Algorithmica*, pages 403–430, 1987.

[5] S.A. NOLFI and D.A. FLOREANO. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. A Bradford book. A BRADFORD BOOK/THE MIT PRESS, 2000.

[6] Wikipedia. Asimo — Wikipedia, the free encyclopedia. `http://upload.wikimedia.org/wikipedia/commons/thumb/0/05/HONDA_ASIMO.jpg/450px-HONDA_ASIMO.jpg`, 2013. [Online; accessed June 23, 2013].

SDSMT SENIOR DESIGN
SOFTWARE DEVELOPMENT AGREEMENT

This Software Development Agreement (the "Agreement") is made between the SDSMT Computer Science

Senior Design Team DanceSoft

Consisting of team members Dicheng Wu and Marcus Berger, ,

and Sponsor Jeff McGough,

with address: _501 E St Joseph St, Rapid City, SD 57701.

1 RECITALS

1. Sponsor desires Senior Design Team to develop software for use by dance studio and dance schools in South Dakota.
2. Senior Design Teams willing to develop such Software.

NOW, THEREFORE, in consideration of the mutual covenants and promises herein contained, the Team and Sponsor agree as follows:

2 EFFECTIVE DATE

This Agreement shall be effective as of October 1, 2015

3 DEFINITIONS

1. "Software" shall mean the computer programs in machine readable object code form and any subsequent error corrections or updates supplied to Sponsor by Senior Design Team pursuant to this Agreement.
2. "Acceptance Criteria" means the written technical and operational performance and functional criteria and documentation standards set out in the DanceSoft Senior Design Documentation
3. "Acceptance Date" means the date of the South Dakota School of Mines and Technology Senior Design Fair, when all Deliverables have been accepted by Sponsor in accordance with the Acceptance Criteria and this Agreement.
4. "Deliverable" means a deliverable specified in the DanceSoft Senior Design Documentation section 1.3
5. "Delivery Date" shall mean, the end of the spring 2017 semester on which University has delivered to Sponsor all of the Deliverables in accordance with section 1.3 of the DanceSoft Senior Design Documentation and this Agreement.

6. “Documentation” means the documents, manuals and written materials (including end-user manuals) referenced, indicated or described in the DanceSoft Senior Design Documentation or otherwise developed pursuant to this Agreement.
7. “Milestone” means the completion and delivery of all of the Deliverables or other events which are included or described in section 1.3 of the Dancesoft Senior Design Documentation scheduled for delivery and/or completion on a given target date; a Milestone will not be considered completed until the Acceptance Date has occurred with respect to all of the Deliverables for that Milestone.

4 DEVELOPMENT OF SOFTWARE

1. Senior Design Team will use its best efforts to develop the Software described in [the project plan.] The Software development will be under the direction of or his/her successors as mutually agreed to by the parties (“Team Lead”) and will be conducted by the Team Lead. The Team will deliver the Software to the satisfaction of the course instructor that reasonable effort has been made to address the needs of the client. The Team understands that failure to deliver the Software is grounds for failing the course.
2. Sponsor understands that the Senior Design course’s mission is education and advancement of knowledge, and, consequently, the development of Software must further that mission. The Senior Design Course does not guarantee specific results or any results, and the Software will be developed only on a best efforts basis. The Software is considered PROOF OF CONCEPT only and is NOT intended for commercial, medical, mission critical or industrial applications.
3. The Senior Design instructor will act as mediator between Sponsor and Team; and resolve any conflicts that may arise.

5 COMPENSATION

No COMPENSATION will occur.

6 CONSULTATION AND REPORTS

1. Sponsor’s designated representative for consultation and communications with the Team Lead shall be Jeff McGough or such other person as Sponsor may from time to time designate to the Team Lead.
2. During the Term of the Agreement, Sponsor’s representatives may consult informally with course instructor regarding the project, both personally and by telephone. Access to work carried on in University facilities, if any, in the course of this Agreement shall be entirely under the control of University personnel but shall be made available on a reasonable basis.
3. The Team Lead will submit written progress reports. At the conclusion of this Agreement, the Team Lead shall submit a comprehensive final report in the form of the formal course documentation at the conclusion of the Senior Design II course.

7 CONFIDENTIAL INFORMATION

1. The parties may wish, from time to time, in connection with work contemplated under this Agreement, to disclose confidential information to each other ("Confidential Information"). Each party will use reasonable efforts to prevent the disclosure of any of the other party's Confidential Information to third parties for

   a period of three (3) years after the termination of this Agreement, provided that the recipient party's obligation shall not apply to information that:

   (a) is not disclosed in writing or reduced to writing and so marked with an appropriate confidentiality legend within thirty (30) days of disclosure;

   (b) is already in the recipient party's possession at the time of disclosure thereof;

   (c) is or later becomes part of the public domain through no fault of the recipient party;

   (d) is received from a third party having no obligations of confidentiality to the disclosing party; (e) is independently developed by the recipient party; or (f) is required by law or regulation to be disclosed.

2. In the event that information is required to be disclosed pursuant to subsection (6), the party required to make disclosure shall notify the other to allow that party to assert whatever exclusions or exemptions may be available to it under such law or regulation.

8 INTELLECTUAL PROPERTY RIGHTS

All intellectual property created for use in the DanceSoft project is property of the South Dakota Board of Regents.

9 WARRANTIES

The Senior Design Team represents and warrants to Sponsor that:

1. the Software is the original work of the Senior Design Team in each and all aspects;
2. the Software and its use do not infringe any copyright or trade secret rights of any third party.

No agreements will be made beyond items (1) and (2).

10 INDEMNITY

1. Sponsor is responsible for claims and damages, losses or expenses held against the Sponsor.
2. Sponsor shall indemnify and hold harmless the Senior Design Team, its affiliated companies and the officers, agents, directors and employees of the same from any and all claims and damages, losses or expenses, including attorney's fees, caused by any negligent act of Sponsor or any of Sponsor's agents, employees, subcontractors, or suppliers.
3. NEITHER PARTY TO THIS AGREEMENT NOR THEIR AFFILIATED COMPANIES, NOR THE OFFICERS, AGENTS, STUDENTS AND EMPLOYEES OF ANY OF THE FOREGOING, SHALL BE LIABLE TO ANY OTHER PARTY HERETO IN ANY ACTION OR CLAIM FOR CONSEQUENTIAL OR SPECIAL DAMAGES, LOSS OF PROFITS, LOSS OF OPPORTUNITY, LOSS OF PRODUCT OR LOSS OF USE, WHETHER THE ACTION IN WHICH RECOVERY OF DAMAGES IS SOUGHT IS BASED ON CONTRACT TORT (INCLUDING SOLE, CONCURRENT OR OTHER NEGLIGENCE AND STRICT LIABILITY), STATUTE OR OTHERWISE. TO THE EXTENT PERMITTED BY LAW, ANY

STATUTORY REMEDIES WHICH ARE INCONSISTENT WITH THE PROVISIONS OF THESE TERMS ARE WAIVED.

11 INDEPENDENT CONTRACTOR

For the purposes of this Agreement and all services to be provided hereunder, the parties shall be, and shall be deemed to be, independent contractors and not agents or employees of the other party. Neither party shall have authority to make any statements, representations or commitments of any kind, or to take any action which shall be binding on the other party, except as may be expressly provided for herein or authorized in writing.

12 TERM AND TERMINATION

1. This Agreement shall commence on the Effective Date and extend until the end of classes of the second semester of Senior Design (CSC 464), unless sooner terminated in accordance with the provisions of this Section ("Term").
2. This Agreement may be terminated by the written agreement of both parties.
3. In the event that either party shall be in default of its materials obligations under this Agreement and shall fail to remedy such default within thirty (30) days after receipt of written notice thereof, this Agreement shall terminate upon expiration of the thirty (30) day period.
4. Any provisions of this Agreement which by their nature extend beyond termination shall survive such termination.

13 ATTACHMENTS

Attachments A and B are incorporated and made a part of this Agreement for all purposes.

14 GENERAL

1. This Agreement constitutes the entire and only agreement between the parties relating to the Senior Design Course, and all prior negotiations, representations, agreements and understandings are superseded hereby. No agreements altering or supplementing the terms hereof may be made except by means of a written document signed by the duly authorized representatives of the parties.
2. This Agreement shall be governed by, construed, and enforced in accordance with the internal laws of the State of South Dakota.

15 SIGNATURES

[illegible] 10/1/2015

Marcus Berger Date

[illegible] 10/1/15

Dicheng Wu Date

Jeff McGough Date

A

Product Description

Write a description of the product to be developed. Use sectioning commands as neccessary.

NOTE: ***This is part of the contract.***

B

Publications

Research Track: This chapter will include any publications generated from the research. Most likely these will be preprints and one will just include the pdf.

C

Sprint Reports

1 Sprint Report #1

Sprint Report #1

0.1 Team Members:

Dicheng Wu
Marcus Berger
Sponsor:
Jeff McGough

0.2 Customer description

0.2.1 Description of sponsoring customer

The sponsoring customer for this project is Dr. Jeff McGough a computer science professor at the South Dakota School of Mines and Technology and the vice president of the Academy of Dance Arts in Rapid City, South Dakota. Well not the sponsor of the project Dr. McGough's wife is also a key part of the customer base as the owner of the Academy. She and other dance school owner are the target group for this project.

0.2.2 Statement of customer's problem or goal for this project

The customer wants a software that can run the day to day operations of a dance studio and also handle record keeping, billing, payroll, and other business operations

0.2.3 Customer's Needs

The customer needs us to develop a software solution which can run the dance studio in an effective manner. The product also needs to handle changing classes from year to year without needing to be updated. This means that the software needs to sync with multiple users, and handle new information such as class rosters, prices, clothing requirements for classes, changes in the employment roster, and many other changes that can occur in the running of a dance school.

This project as a whole needs to be an improvement on the current system in use by the customer and provide an easy and efficient way to run the clients business.

0.3 Overview of the project:

The project consists of three major parts, Gui, database and back end code. For the Gui part, we are going to integrate everything into a small number of windows to eliminate the need for large numbers of window like the current product in use at the academy. and, thus, users can manipulate it very straightforwardly. For database part, we are going to build a database which stores students, employees, classes, billing, and other information needed for the academy to function as a business. For the code back end, we are going to develop it on using Python and PyQt with a primary focus on Mac but with cross platform compatibles.

0.4 Project Environment:

0.4.1 Project boundaries

The boundaries of this project would be the Academy of Dance Arts in Rapid City. The product could be used by other dance schools in the future but they are out of the scope of this projects development

0.4.2 Project context

The context of this project is only to develop a better software solution to the current software in use at the Academy of Dance Arts in Rapid City, South Dakota, so they can run their school in a more efficient manner. While the project is open source it is not the intention of this team to develop an all purpose solution for all the dance schools around the country. This project is tailored to the needs of the Academy of Dance Arts.

0.5 Project deliverables of Sprint 1:

1. The research into program languages, database and Gui frameworks and architectures for the DanceSoft project.
2. Final decision on frameworks and architectures for the DanceSoft project.
3. User Stories and Product Backlogs for the project.
4. Creating a very simple Qt window.

0.6 User Stories

After the requirement for the project were laid out the team created the user stories based on those requirement. The user stories the team came up with are as fallows:

1. As a user i want to adjust students payment models
2. As the owner I would like to see automatic database backups.
3. As a student I would like to be able to register online (with special app). Classes must be approved before added.
4. As a student I would like to be able to search clothing requirements.
5. As a student I would like to know my billing.
6. As the owner I would like to indicate clothing requirements per class.
7. As a studio person, I would like to be able to add students to classes.
8. As a student, teacher etc, I would like to be able to look up a students class list.
9. As the teacher I would like to get a class role for each class.
10. Given a class list, I would like to get an invoice of the tuition due.
11. Studio would like to track payments and estimate remainder due. I would like to generate an invoice for this amount.
12. As a student I would like to be able to register online (with special app). Classes must be approved before added.
13. As a student I would like to know my billing.

14. As the owner I would like to track teacher hours and compute payroll.
15. As the owner I would like to indicate clothing requirements per class.
16. As a student I would like to be able to search clothing requirements.
17. As the owner I would like to see automatic database backups.

0.7 Product Backlog:

From the above user stories the team produced the fallowing product backlog. The parts are not in the order of execution. More thing will be add or removed to this in the future as the software develops.

1. Create the database tables
2. Create database backup system
3. Create online register page and approval system
4. create clothing requirement update function, and add clothing function for teachers and admin
5. create clothing viewer function for students
6. create dynamic billing algorithms (expand on later sprint)
7. create page for tuition calculation and viewing by students
8. Add and subtract students from a class
9. Create dynamic Queries to produce class attendance list
10. Hour tracker for teachers and payroll calculation algorithm (expanded later)
11. Using dynamic DNS service to set up Linux Box
12. Encrypt data that store in the database
13. retrieve data from database to produce a invoice for payroll
14. queries in database to retrieve student information
15. Create ability for employee to look up and modify student registration info
16. Query database to produce class role sheet

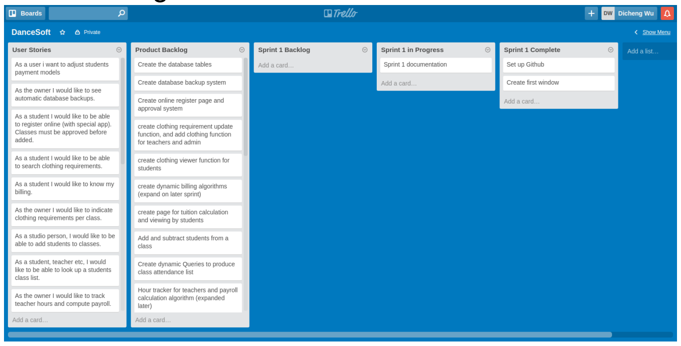


Figure 1: DanceSoft Trello Board

DanceSoft

User Stories
- As a user i want to adjust students payment models
- As the owner I would like to see automatic database backups.
- As a student I would like to be able to register online (with special app). Classes must be approved before added.
- As a student I would like to be able to search clothing requirements.
- As a student I would like to know my billing.
- As the owner I would like to indicate clothing requirements per class.
- As a studio person, I would like to be able to add students to classes.
- As a student, teacher etc, I would like to be able to look up a students class list.
- As the owner I would like to track teacher hours and compute payroll.

Product Backlog
- Create the database tables
- Create database backup system
- Create online register page and approval system
- create clothing requirement update function, and add clothing function for teachers and admin
- create clothing viewer function for students
- create dynamic billing algorithms (expand on later sprint)
- create page for tuition calculation and viewing by students
- Add and subtract students from a class
- Create dynamic Queries to produce class attendance list
- Hour tracker for teachers and payroll calculation algorithm (expanded later)

Sprint 1 Backlog

Sprint 1 in Progress
- Sprint 1 documentation

Sprint 1 Complete
- Set up Github
- Create first window

17. Query database to produce employee information
18. create permission assignment system
19. handle billing info (expand later)
20. Create dynamic algorithm for payment model creation and calculation.
21. Algorithm for payment tracking and remainder calculation, and query database to produce invoice.
22. Query and insert information needed to create and new class and produce a results page.
23. Create update query to assign teachers to a class

0.7.1 Sprint 1 Backlog:

1. Set up Github
2. Design Research
3. Design Decision
4. Create first window
5. Sprint 1 documentation
6. Sprint 1 Review
7. Continuing practice with QT

0.8 Research

0.8.1 Database Research

SQL VS NonSQL:

SQL is designed for fixing data structure and the scale of amount of data in database should be medium size otherwise with it growing big the database will drop down its speed. NonSQL is designed for frequent changing data structure and the scale of amount of data in database does not affect the performance of the database a lot. The SQL database is stable and however the NonSQL is constantly suffering failovers. The database aims to store employees and students and classes information and the size of students less than 4000 and the size of employees less than 10. The structure of data is stable, by considering all those facts above, we decide to use SQL database.

The different SQL Databases:
For this part, we think of using Mysql, SqLite, Mssql. Mysql is a free software and widely used in different fields. It has a very good portability and can runs over different OS systems and aims for small size of data. Also our experience is more focused in MySQL, and the feature set for SQL in MySQL fits within the scope of the project, for these reasons we choose MySQL as the database frame work.

0.8.2 Storage Options:

For this part, We considered three options, local, cloud, mixed. Since the database is accessed by different devices and the internet in the Academy is not stable, we decide to use mixed storage schema either Amazon AWS plus a local database or a Linux box plus a local database. After researching Amazon AWS, it does not support storing images and videos which the Academy may require in the future, therefore due to some inconsistency in the internet and the current and possible future needs of the Academy we decided to go with a local Linux box and database.

0.8.3 Languages

Since the Academy is currently running a Mac system the first language idea was objective-C. Recently though Apple released a programming language update called Swift. Due to our teams inexperience with Mac and the possibility of the dance academy being sold in the future we were faced with a choice, between Python or Swift. Swift is a c++ like language which stuck out as a possible jumping off point for us. However as we researched it became clear that swift would have a high ramp up time and learning curve. On top of this as novices to the Mac development environment we would spend a large amount of time just trying to figure out the system. As far as pluses for Swift the language has all the functionally of objective-C plus things added to make the language better, overall reception for the language within the Mac community have been positive. The language itself is designed with porting to IOS in mind which would make a mobile transition more likely. The language would use Cocco as a GUI environment which is also relatively well revived by OSX developers, ut again ramp up for our team would be high.

The language competing with Swift in our discussions was Python. Python has the upside of being a language the team is more experienced in. Also the client Dr. McGough knows Python so any updates would be easier for him to do. Python is also a cross platform language which allows the team to produce working code for Mac, Windows, and Linux at the same time and on any operating system. This means the the team can develop on Windows or Linux, development environments we are more familiar with, and have the code work on Mac. Also development in Python means that if the academy was ever sold to a Windows user the product would still work. Another advantage we discovered to python is the academic and career experience it provide since in our research for the SD Mines career fair we discovered that many companies use Python and more than expected would like QT experience. Lastly the Python community is larger and can more readily provide assistance if needed through websites and research.

So overall while Python is not Mac native it provides more viable reasons for use in our teams eyes. That is not to say we don't believe Swift is a good choice, Swift is a completely viable choice for a product like this it is just not the best for the exact situation the team is in. So we have decided to create the Academy of Dance Arts Software in Python using PyQt as a GUI.

0.8.4 Final Framework Decision

Language: Python
Gui: PyQt
Database: Mysql

0.9 Foreseeable issues

As of the sprint 1 review, possibly issues could include database encryption and synchronization, team learning curve of Qt.

2 Sprint Report #2

Sprint Report #2

0.1 Team Members:

Marcus Berger
Dicheng Wu
Sponsor:
Jeff McGough

0.2 Prototype Progress

The progress is mostly out of the research phase and has began the first steps of production. The progress is laid out in more detail in this report. As of now the prototype is in a state to begin working on functionality. The rough (not final) GUI pages are laid out and being constructed to give us an environment for creating the functionality. The back end database has been constructed and will allow us to begin testing the database connected page as we create them. Current pages constructed are log in page, landing pages, and some options page.

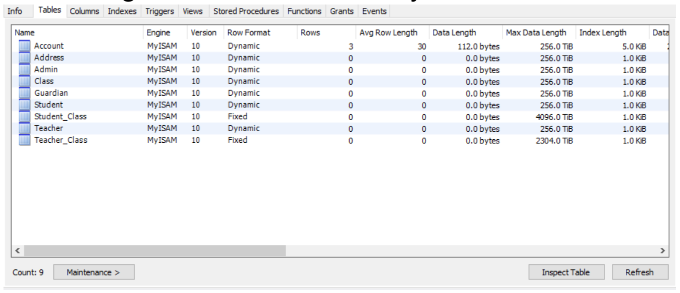
0.3 Project deliverables of Sprint 2:

1. The database creation script
2. Gui path work, meaning figuring out the path through the Gui
3. Log in page that reads users permission level and sends them to the correct landing page
4. Rough landing pages for Admin and Teachers (design improvements to come in later sprint)

0.3.1 Sprint 2 Backlog:

1. Creation of database tables
2. Draw Gui path

Figure 1: Tables currently in database



Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length
Account	MyISAM	10	Dynamic	3	30	112.0 bytes	256.0 TiB	5.0 KiB
Address	MyISAM	10	Dynamic	0	0	0.0 bytes	256.0 TiB	1.0 KiB
Admin	MyISAM	10	Dynamic	0	0	0.0 bytes	256.0 TiB	1.0 KiB
Class	MyISAM	10	Dynamic	0	0	0.0 bytes	256.0 TiB	1.0 KiB
Guardian	MyISAM	10	Dynamic	0	0	0.0 bytes	256.0 TiB	1.0 KiB
Student	MyISAM	10	Dynamic	0	0	0.0 bytes	256.0 TiB	1.0 KiB
Student_Class	MyISAM	10	Fixed	0	0	0.0 bytes	4096.0 TiB	1.0 KiB
Teacher	MyISAM	10	Dynamic	0	0	0.0 bytes	256.0 TiB	1.0 KiB
Teacher_Class	MyISAM	10	Fixed	0	0	0.0 bytes	2304.0 TiB	1.0 KiB

Count: 9

3. Decision on rough Gui theme
4. Create a log in page that sends the user to the correct landing page based on their permission level
5. Create rough versions of teacher and admin landing pages to test functionally (improve look in latter sprint
6. Continue rough gui page creation to have environments for functionality
7. Sprint 2 Review
8. Continuing practice with QT

0.4 Database Creation

The first thing we tackled in sprint 2 was getting the database up so we could begin actually developing the product and give our qt interface something to actually interface with. The main goal here was to make sure we had constructed the database in such a way that it could complete all the user stories it needed to in a way that made sense. After talking through the tables and the users stories we came up with the table creation script that was submitted with this sprint. While modifications will need to be made for when we do the billing side of the project, we think that this table structure should provide for the need of the academy.

A few thing to note in the database, we created student_class and teacher_class tables to deal with the many to many relationships in the database. Also there are no payroll or transaction tables yet as the group is still trying to figure out how to handle billing information and data.

0.5 GUI Work

0.5.1 GUI Path and Theme

The path for the gui was part of the backlog so we could figure out the minimum number of pages we would need to do the things required by this project. This does not mean we are looking to create only the minimum but just get a general idea of how many pages we would need. Also it allowed us to see how the pages were connected and how we expect navigation through the system to work. The rough drawing of the path can be found in the Github repo's sprint two folder.

The other thing we need was to talk to the client about what he wanted for a navigation theme. We came up with three options, first was a button layout on the top and side of the page, second was drop down menu similar to the way was mac and windows handle navigation in their products. Last was an expanding folder structure similar to that of the content management system Ektron. After consulting with Dr. Mcgough, he decided that the more familiar drop down menu would be the easiest and most user friendly option and the option he would prefer we do. This decision allows us to use the built in menu bar functionality of pyqt and should reduce work load on navigation for this project.

Another part of theme is color and design, the color scheme will most likely resemble the academy's color plate, but the improve design will be worked on later. As of now the primary focus is making sure all the functionality works.

0.5.2 GUI Pages

The GUI pages for this sprint were the log in page and the landing pages for admin and teachers. The student part of the gui will be php which will be worked on in a future sprint once the functionlity is done. Since the student functions are similar to some of the admin or teacher functions most work should port over fairly quickly.

The first page we made was a login in page that takes a username and a password and first checks to see if they exist and are correct within the system. If they are not a dialog saying whats wrong appears and prompts the user again. If the information passes the check the system reads the users permission level and sends them to the corresponding landing page.

Figure 2: Current iteration of the log in page

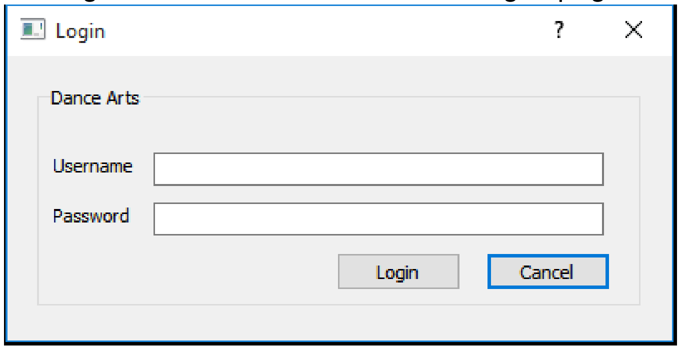
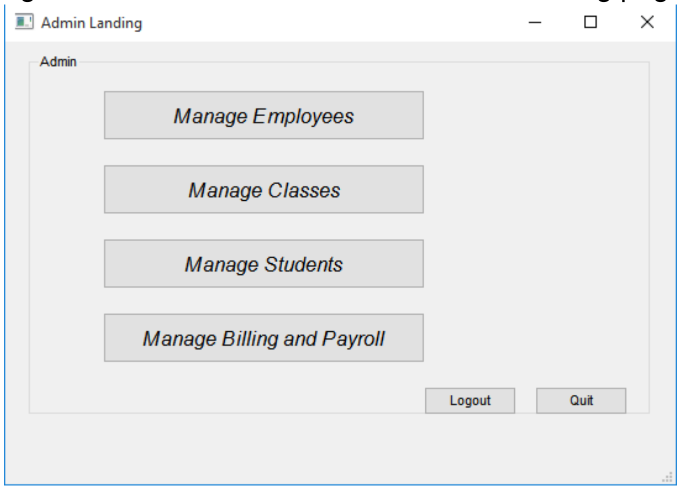


Figure 3: Current iteration of the Admin Landing page



The other set of pages we need to make to begin writing functionality was the landing pages for the admin and teacher permission levels. These pages show the types of things each can do and allow navigation to the various different functionality that permission level can do. For example the admin landing page has a button to take you to a student options page, from there you select which option you want and will be taken to the page to execute that function. The idea is to start at the landing page and be able to navigate to a specific function in three or four clicks max. The menu bar will also execute navigation and should allow the user to jump to a desired function.

Pages currently being worked on include:

1. Search pages
2. Modify data page
3. Add information and new information pages

0.6 Sprint 2 Issues

Some issues that were encountered during this sprint were:

1. Running out of time - the group found it hard to complete this sprint on time due to many time draining issues, such as other class work,family matters, midterm exams and a client presentation during this sprint. Mitigation for this issue will be better time management for the team.
2. Inexperience with pyqt - The team ran into some issues that drained on time when working with pyqt, such as looking up information on how to perform a needed task. Mitigation for this issue is mostly experience based as we learn the system and learn how to more efficiently navigate the community the time sink should go down.
3. Team communication - The team found some issues in communicating with one another because of the language barriers within the team. Which in some cases slowed sprint progress. Mitigation for this issue will come with time as the group begin to understand each other better and find efficient patterns of communication.

0.7 Client Interactions

Client interactions came in the form of weekly meeting every wednesday at 2:00 p.m. Topics included:

1. Progress reports on where we are in the project
2. Asking questions to refine functionality and understand academy needs
3. Discuss the future of the project and idea on how to execute them

0.8 Group Meeting

The group has a standard meeting time of 4:00-5:00 or 4:00-6:00 MWF and a second meeting time of TTH 10:00 - 12:00

Meeting can continue pass these times if needed, and other times during the weekend. However weekend times fluctuate and do not remain constant from week to week.

0.9 Work Distribution

Marcus:

1. Gui Path Charts
2. Landing page Construction

Dicheng:

1. GUI Theme Ideas
2. Log In Page and Permission Navigation

Together:

1. Wrote and talked through database and construction
2. Got database running
3. GUI page breakdown based on user stories and product backlog
4. Coded some functionality

3 Sprint Report #3

Sprint Report #2

0.1 Team Members:

Marcus Berger
Dicheng Wu
Sponsor:
Jeff McGough

0.2 Prototype Progress

The progress is being made on the simpler functions of the projects including search, updates, and role sheets. As of now the prototype is in a state of some working functionality on the admin and teacher side. The pages are not yet tied together but the complete functions function independently. Several GUI pages are complete or in a working state to compliment these functions. The back end database has been slightly updated to adjust to the needs of the project. Lastly the prototype has reached a state where testing can be conducted on parts of the project, more information is given later in this document.

0.3 Project deliverables of Sprint 3:

1. Student search page
2. Employee search page
3. Add a class to the database
4. Update teacher and student information pages
5. Modify clothing requirements for a class
6. Generate a class role sheet
7. Assign teacher to a class
8. Ability to add and subtract students from a class

Figure 1: Search Pages

0.3.1 Sprint 3 Backlog:

1. Queries in database to retrieve student and employee information
2. Query and insert information needed to create and new class and produce a results page.
3. Query database to produce class role sheet
4. Create clothing requirement update function, and add clothing function for teachers and admin
5. Create update query to assign teachers to a class
6. Create ability for employee to look up and modify student registration info
7. Add and subtract students from a class
8. Sprint 3 Review
9. Create Client presentation
10. Documentation for semester

0.4 Search Pages

The first page tackled this sprint was the search page. The page takes user input and searches based on name using a fuzzy search. Also included in both searches is an advanced search option that allows the user to check boxes corresponding to the fields in the database, which allows the user to search in more versatile ways. Lastly the user can click on a student to pull up all of their information and modify it as needed.

0.5 Add A Class

Clicking on the "Add a Class" button on the admin landing page will open a dialog. From this dialog box the user can type in information to a add a class. These fields include class name, cost, start and end times, day, clothing requirements, class description, start and end dates, and the age range for the class. At the time of class creation only name, is a required field. When the submit button is clicked a message box will pop up to confirm the database submission before submitting to the database

0.6 Role Sheet

This page generates a list of classes teachers are currently teaching. Teacher can see who is taking his/her class by selecting the corresponding class on the list and can print this list as a pdf.

0.7 Update Pages

These pages are fairly simple to explain sometimes the user will need to add or alter a record of some kind these pages need to be able to do that in a simple and concise way. The update pages worked on in this sprint include student information from the employee side, updating teacher information and updates to clothing requirements for a class. Most of these page will just produce a form where information can be displayed and updated.

0.8 Assign Teacher to a Class

This page allows an admin to select a class, clicking on a class pulls up a list of available teachers to select to teach the class. Clicking on a teacher will pull up a message box to confirm the selections before submitting to the database. The available teachers are selected based on the start time of the selected class and the end time of the classes already being taught by each teacher. If the class times overlap then that teacher will not show up in the available teachers list.

0.9 Sprint 3 Issues

Some issues that were encountered during this sprint were:

1. Time - While not as much of a problem as it was in sprint two the group still found itself running low on time. The lead gained during Christmas break should reduce the issue in phase 2.
2. Inexperience with pyqt - While still a small issue the experience of the team is growing and while some issues still exist such as needing to look up information at times, the issues are lessened from the last sprint.
3. Team communication - The team found some issues in communicating with one another because of other projects and assignments. Which in some cases slowed sprint progress. Mitigation for this issue will be to more strongly impose a schedule in phase 2.

0.10 Client Interactions

Client interactions came in the form of weekly meeting every Wednesday at 2:00 p.m. Topics included:

1. Progress reports on where we are in the project
2. Asking questions to refine functionality and understand academy needs
3. Discuss the future of the project and idea on how to execute them

0.11 Group Meeting

The group has a standard meeting time of TTH 10:00 - 12:00 and once during the weekend usually from 1:00-5:00

Meeting can continue pass these times if needed, and other times during the weekend. However extra weekend times fluctuate and do not remain constant from week to week.

0.12 Work Distribution

Marcus:

1. Add a class

2. Assign teacher to class
3. Update teacher page
4. Update clothing
5. documentation
6. trello management

Dicheng:

1. Search Pages
2. Role sheet
3. Modify student information
4. Add and Subtract students from a class
5. Navigation to teacher or admin page

Together:

1. Updated database construction
2. GUI page breakdown based on user stories and product backlog
3. Coded some functionality

4 Sprint Report ...

D

Industrial Experience and Resumes

1 Resumes

Marcus Berger

4534 Bozeman Cir. Rapid City, SD 57703 605-430-2940
marcus.berger@mines.sdsmt.edu

Employment

Intern, SDSM&T University Relations**,** Rapid City, SD 9/2013 to 8/2015

- Managed and created website content utilizing HTML and style sheets
- Experience with Content Management Systems (Estrada, Ektron)
- Marketing Exposure through SDSM&T PR campaigns and programs

Computer Landscape Designer, Turf and Erosion Solutions**,** Rapid City, SD 5/2013 to 8/2013

- Created and set up landscapes on computer for customers
- Created mock ups to help customer visualize projects such as large company buildings

Education

Bachelor of Science in Computer Science Anticipated: 5/2016
South Dakota School of Mines and Technology, Rapid City, SD GPA 2.85

- **Courses including but not limited to:** Finite Structures, Data Structures, Analysis of Algorithms, Database Management, Assembly Language, and Computer Graphics
- **Current Major Courses:** Networking, Data Mining, AI, GUI, OS, and Senior Design
- **Projects include:** Data Manipulation, Image Processer (C++ and ARM Assembly Language), Hamming Code Simulation, Solar System Model in OpenGL, Database Projects including PHP and SQL
- **Senior Design Project:** Developing an open source program, database and GUI to handle student information and registration, and other business information for a local dance school

Qualifications

Professional Skills:	Software Skills:
Experience with Agile development and Github	Fluent in C++
Knowledge of Waterfall development	Proficient in Python
Task Driven Development	Experience with HTML, PHP and SQL
Working in Teams	Some Lisp
Oral and Written Communication	Webpage Design (CMS, Google Analytics, etc.)

References – Available Upon Request

2 ABET: Industrial Experience Reports

2.1 Marcus Berger

2.2 Dicheng Wu

E

Acknowledgment

The team would like to acknowledge the following people from their input and assistance on this project:

1. Brian Butterfield - For assistance in senior design class and project input
2. Dr. Mengyu Qiao - For assistance with database construction and security options
3. Computers Unlimited Senior Design Group - For sharing the mobile lab and meeting times every week without fail.
4. Fellow Members of the Senior Design Class - For input, recommendations, and assistance during project development

F

Supporting Materials

This document will contain several appendices used as a way to separate out major component details, logic details, or tables of information. Use of this structure will help keep the document clean, readable, and organized.

LaTeX Example

LaTeX sample file: Remove from submitted materials

1 Introduction

This is a sample input file. Comparing it with the output it generates can show you how to produce a simple document of your own.

2 Ordinary Text

The ends of words and sentences are marked by spaces. It doesn't matter how many spaces you type; one is as good as 100. The end of a line counts as a space.

One or more blank lines denote the end of a paragraph.

Since any number of consecutive spaces are treated like a single one, the formatting of the input file makes no difference to TeX, but it makes a difference to you. When you use LaTeX, making your input file as easy to read as possible will be a great help as you write your document and when you change it. This sample file shows how you can add comments to your own input file.

Because printing is different from typewriting, there are a number of things that you have to do differently when preparing an input file than if you were just typing the document directly. Quotation marks like "this" have to be handled specially, as do quotes within quotes: "'this' is what I just wrote, not 'that'".

Dashes come in three sizes: an intra-word dash, a medium dash for number ranges like 1–2, and a punctuation dash—like this.

A sentence-ending space should be larger than the space between words within a sentence. You sometimes have to type special commands in conjunction with punctuation characters to get this right, as in the following sentence. Gnats, gnus, etc. all begin with G. You should check the spaces after periods when reading your output to make sure you haven't forgotten any special cases. Generating an ellipsis . . . with the right spacing around the periods requires a special command.

TeX interprets some common characters as commands, so you must type special commands to generate them. These characters include the following: $ & % # { and }.

In printing, text is emphasized by using an *italic* type style.

A long segment of text can also be emphasized in this way. Text within such a segment given additional emphasis with Roman *type. Italic type loses its ability to emphasize and become simply distracting when used excessively.*

It is sometimes necessary to prevent TeX from breaking a line where it might otherwise do so. This may be at a space, as between the "Mr." and "Jones" in "Mr. Jones", or within a word—especially when the word is a symbol like *itemnum* that makes little sense when hyphenated across lines.

Footnotes[1] pose no problem.

[1]This is an example of a footnote.

TEX is good at typesetting mathematical formulas like $x - 3y = 7$ or $a_1 > x^{2n}/y^{2n} > x'$. Remember that a letter like x is a formula when it denotes a mathematical symbol, and should be treated as one.

3 Displayed Text

Text is displayed by indenting it from the left margin. Quotations are commonly displayed. There are short quotations

> This is a short a quotation. It consists of a single paragraph of text. There is no paragraph indentation.

and longer ones.

> This is a longer quotation. It consists of two paragraphs of text. The beginning of each paragraph is indicated by an extra indentation.
>
> This is the second paragraph of the quotation. It is just as dull as the first paragraph.

Another frequently-displayed structure is a list. The following is an example of an *itemized* list.

- This is the first item of an itemized list. Each item in the list is marked with a "tick". The document style determines what kind of tick mark is used.
- This is the second item of the list. It contains another list nested inside it. The inner list is an *enumerated* list.
  1. This is the first item of an enumerated list that is nested within the itemized list.
  2. This is the second item of the inner list. LATEX allows you to nest lists deeper than you really should.

  This is the rest of the second item of the outer list. It is no more interesting than any other part of the item.
- This is the third item of the list.

You can even display poetry.

> There is an environment for verse
> Whose features some poets will curse.
>
> For instead of making
> Them do *all* line breaking,
> It allows them to put too many words on a line when they'd rather be forced to be terse.

Mathematical formulas may also be displayed. A displayed formula is one-line long; multi-line formulas require special formatting instructions.

$$x' + y^2 = z_i^2$$

Don't start a paragraph with a displayed equation, nor make one a paragraph by itself.

4 Build process

To build LATEX documents you need the latex program. It is free and available on all operating systems. Download and install. Many of us use the TexLive distribution and are very happy with it. You can use a editor and command line or use an IDE. To build this document via command line:

```
alta>  pdflatex SystemTemplate
```

If you change the bib entries, then you need to update the bib files:

```
alta>  pdflatex SystemTemplate
alta>  bibtex SystemTemplate
alta>  pdflatex SystemTemplate
alta>  pdflatex SystemTemplate
```

The template files provided also contain a Makefile, which will make things much easier.

Acknowledgment

Thanks to Leslie Lamport.