DanceSoft

Senior Design Final Documentation

DanceSoft

Marcus Berger   Dicheng Wu

April 26, 2016

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
12/9/15	Marcus Berger	1.0.3	Updated Requirements (future user stories marked with update time frame)
12/9/15	Dicheng Wu	1.0.3	Uploaded Resume
12/9/15	Marcus Berger	1.0.3	Minor Update to Testing
12/10/15	Dicheng Wu	1.0.3	Updates to Design and Develop
12/10/15	Marcus Berger	1.0.3	Product Contract Description
3/25/15	Marcus Berger	1.0.3	Merged over to new design template
3/30/15	Marcus Berger	1.0.5	Reiterate the overview section and fill out the experience and bibliography

1

Overview and concept of operations

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

1.1 Team Members and Team Name

The DanceSoft Team consist of:

1. Marcus Berger

2. Dicheng Wu

1.2 Client

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 and artistic director of the Academy of Dance Arts in Rapid City, South Dakota. The clients main goal is to eventually use the software after some iterations to manage the day to day activities of the Academy, and as the number of iterations of the project increase add student interfaces and refines the project to a point of business viability and security.

1.3 Project

This first iteration of the project is a proof of concept for the DanceSoft project for purposed use at the Rapid City Academy of Dance Arts. The project after this iteration is meant to be a minimum viable product to give the client as proof of project viability. After this project it will be up to the client, Jeff McGough, to decide whether to keep the project, look elsewhere, or hand over the project to a following senior design team.

As stated above the project is meant to be a minimum viable project, or proof of concept. The system general purpose is running the day to day administrative duties of the Academy of Dance Arts. These duties can range from, but are not necessarily limited to: enter student and class information, payments and billing, or managing employees information.

1.3.1 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 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. The project looks to accomplish this purpose through a local graphical interface and a back end mySQL database for the storage of the data.

1.4   Business Need

The customer needs the team to develop a software solution which can run the dance academy 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 various information, 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 the 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 accomplish the data manipulation task through the back-end MySQL database, and the ease of use will be handled with a simple PyQt interface. The academy also need the system to work will the employees of the academy which can be devided into admin and teacher categories within the system. The team accomplishes this through a log in system to different landing pages which contain the various functionality completed in this first iteration of the project, as laid out in the user stories listed in this document.

1.5   Deliverables

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

1.5.1   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 for this first iteration of the software.

1.5.2   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 the requested operations and functions.

1.6   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 and admins 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. 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.

1.7   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 application after client approval or more iterations should reach a final goal of being deployed on the local machines within the Rapid City Academy of Dance Arts. When a local GUI user connects the user will navigate through the various pages listed above to the desired functionality. Figure 1.1 shows a simplistic view of the GUI Architecture. There are two major components to this iteration the project, database, and admin/teacher interface. Each section is described in more functional detail above and in **section 4 Design**
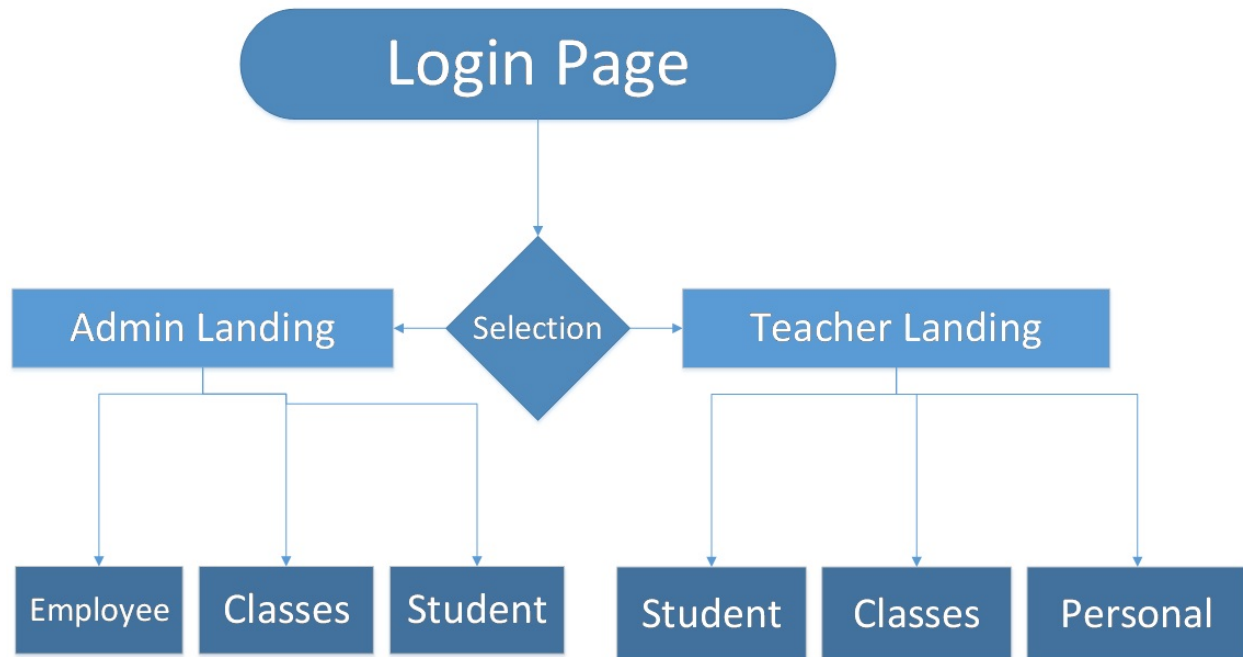
Figure 1.1: Main GUI Overview

and Implementation . The database architecture is a system of mySQL tables connected through the use of various keys. The tables correspond to the necessary information for various functionality within the system. [MB]

1.8   Technologies Overview

The primary Technologies for this projects are as follows:

1. Xcode and Visual Studio - Xcode, Microsoft Visual Studio 2015, and Python IDE were the primary development IDEs for this project. Of the three the ones the group used the most were Visual Studio 2015, and the Python IDE so the team could develop the project on the computers provided by the South Dakota School of Mines and Technology. Xcode test were done every so often to confirm the files worked cross-platform.

2. Python - the primary programming language for the project.

3. PyQt and Qt Designer - GUI package and development environment, these tools are publicly available and can be downloaded for free off the internet.

4. MySQL - MySQL provides the database and relational quires to manage the data and organize it within the system. This software is also free to use and can be downloaded from the MySQL website.

These technologies were selected after a first research sprint where research into programming language, GUI, and database options was conducted and the technologies 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]
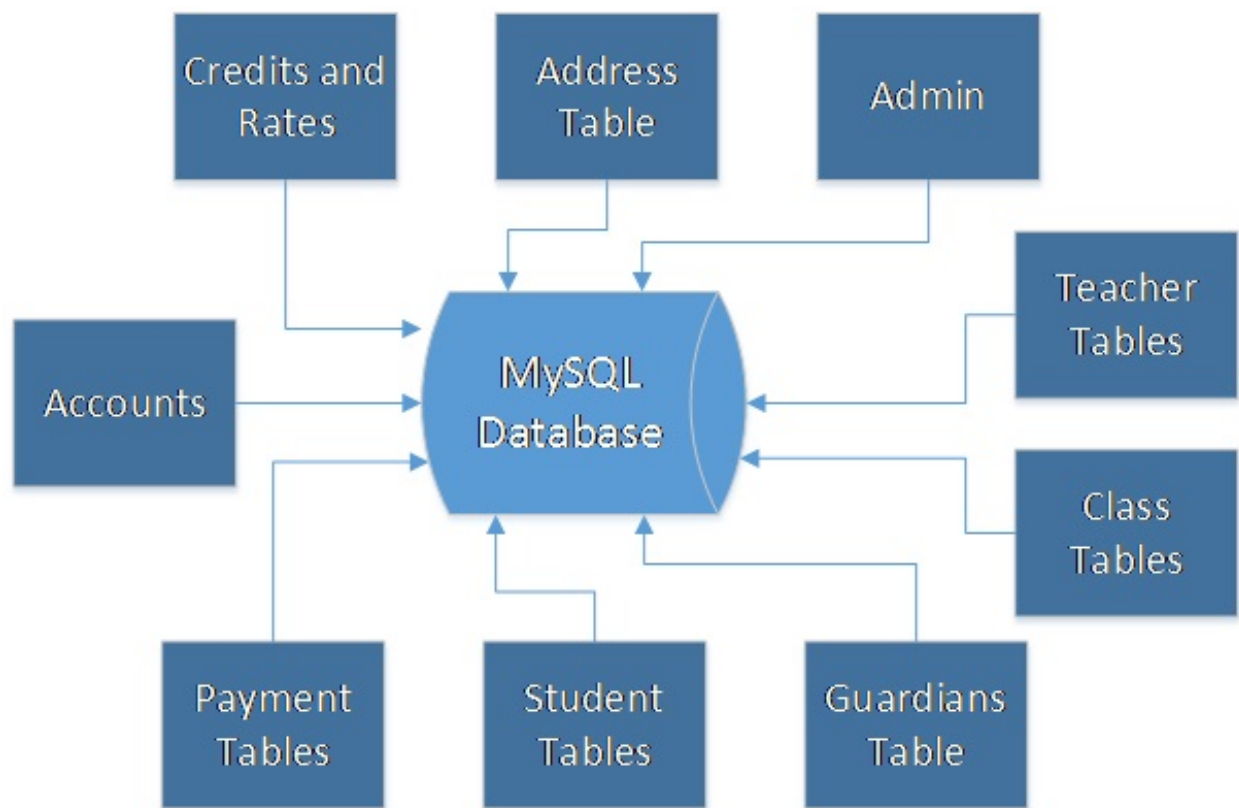
Figure 1.2: Main Database Overview

2

User Stories, Requirements, and Product Backlog

2.1 Overview

This document covers the client information, an overview of the requirements of the system and the requirements laid out by the client. These requirements are laid out in this document with how the group plans to implemented them within the system. [MB]

2.2 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: [MB]

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

2.2.1 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 pay model to another existing payment model. Adding a payment model is part of a different user story.

2.2.1.a User Story #1 Breakdown

Further breakdown for this user story: this functionally was further broken down into, allowing the student payment to be accept in many ways. Ways to make payment include credit, cash, check, and other. During the first iteration of this project this user story was partially completed. The user is able to process a students payment type and log the payments in a payment history.

2.2.1.b User Story #1 Remaining

This user story did not reach full completion in this iteration of the project. The discount types that the academy uses exist within the database, however due to issues with the project the current iteration never reached the point of handling the payment models with in the interface. If next iteration proceeds directly from this iteration the following still need to be implemented:

1. handle the payment model GUI page

2. check to see if a student is on a certain payment model

3. handle discount changes in the database

2.2.2 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.

2.2.2.a User Story #2 Remaining

The DanceSoft team during this iteration of the project did not reach this user story before the submission deadline. The database will most likely be backed up on a machine provided by the client in the next iteration as the project becomes more refined and closer to business deployment.

2.2.3 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.

2.2.3.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.

The team has created a version of this registration approval function, where a admin in the system can approve, or reject a students request to register. Also in this interface the admin can choose to refund the

student if they are dropping the class or just strictly remove the student if they don't need a refund or the class has not yet started.

After the first three sprints the student interface was dropped from this iteration of the project. Future iterations should be able to integrate the student web portal with this system due to the fact that the initial design for this project included considerations for what would be needed in the student interface.

2.2.3.b   Further Breakdown

This perceptive interface should also include the options to manage the student information handled by the system. This include the registration functionality, views, updates, registrations, and submits. The interface will run the requirements of the student and their guardians, most of which are covered by other user requirements.

2.2.4   User Story #4

As a student I would like to know my billing. This should allow the users of the local interface to see what a student still owes and print a log of this information.

In the next project iteration this functionally should also cross over to the student interface. Within the student interface a given student should be able to see things like what they have left to pay, how much they have paid, when the next payment is due, and other pertinent information to the students billing.

2.2.5   User Story #5

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 using an update form. This is accomplished in the system through a add class form where the admin can create a new class and indicate clothing requirements for that class. Also if the user wants to update clothing requirement they can use a class search feature to find and update the necessary requirements.

2.2.6   User Story #6

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 or approval by teachers.

In a future iteration something akin to this functionally will also need to be added to the future student interface to allow student to register through the online portal.

2.2.7   User Story #7

As a teacher or admin 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. This will be accomplished through an output dialog that pops up to display the students schedule and the pending registration interface.

2.2.8   User Story #8

As the teacher I would like to get a class role for each class. Users need to be able to select a class and see who is enrolled in it. Also the list needs to be printable so teachers can take role at a class. This is done through a window where the teacher can select one of the classes they are teaching and print the role sheet for that class.

2.2.9   User Story #9

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.

2.2.9.a   User Story #9 breakdown

Currently this is accomplished through using the tuition rates logged in the database by the user, which were created using the academy payment models on their website. The hours a student in enrolled in are calculated to generate an invoice.

2.2.9.b   User Story #9 remaining

This interface will need to be expanded and adapted as the next iteration modifies or changes the interface structure or mechanics.

2.2.10   User Story #10

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

Based off of the amounts calculated by the interface in user story 10 the student can submit a payment and the employee of the academy can see and print remaining due.

2.2.10.a   User Story #10 remaining

This interface will also need to be expanded and adapted as the next iteration modifies or changes the interface structure or mechanics.

2.2.11   User Story #11

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.

2.2.11.a   User Story #11 breakdown

The team was able to create different pay names and pay rates for each individual employee to generate a wage for the teacher. The teacher will then be able to see the hours logged. The admin can also change the pay rates for different pay names such as driving or off-site rates.

Teachers and employees are also given a way to submit hours through their interface that can then be viewed by the owner.

2.2.11.b   User Story #11 remaining

During the first iteration of the team was able to complete the framework for generating an employees gross wage. The tax calculations will need to be added and adapted in a future iteration of this project.

2.2.12   User Story #12

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.

This is done through an update form where the user can modify the information and submit the updates to the database.

2.2.13   User Story #13

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

2.2.13.a   User Story #13 Breakdown

Search and view functions 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.

2.2.13.b   User Story #13 Remaining

In future iterations this functionality will also be added and adapted for the future student web portal.

2.2.14   User Story #14

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 add a class to the academy's roster. Which is done through an add information form which checks required information and submits it to the database.

   This information can later be viewed and updated through a class search and update function. While assign teacher to classes is done through a separate list view function

2.2.15   User Story #15

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.

2.2.15.a   User Story #15 Remaining

The team during this iteration of the project was able to create the framework in the database necessary for this user story, however the team was unable to impliment this functionality in the interface during this iteration.

2.2.16   User Story #16

As a admin I want to be able to set level of permissions with information.

   This user story is the basis for the log in system where the user can create a teacher, with default level permission. Then add the user as an admin if they choose to. The user can also remove admin complete from the system or just that employees admin credentials.

2.3   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. [MB]

2.3.1   System Requirements

The system requirements laid out by the clients are the necessary features laid out in the user stories above. 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.

   These requirements will carry over in general to any future iterations of this project as other iterations continue to refine, adapt, and rework the various functions of the project.

2.3.2 Network Requirements

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

When the student interface is integrated into the overall projects the network requirement may change or need to adapted through iterations.

2.3.3 Development Environment Requirements

The academy runs on a Mac currently so the system must work on the Mac OSX operating system upon completion. While not required the project is developed in Python, so the end product will work cross platform. So that if the academy ever switches operating systems or if the academy is ever sold the system will work if the new owners run windows. If they choose to use it. Currently the client will store this iteration of the project in a local Linux box, which the client will provide to the team at the time of data transfer.

2.3.4 Project Management Methodology

The client requires a weekly meeting every Wednesday at 2:00 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 in an acceptable iteration 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 for check in purposes.

2.4 Product Backlog

The following is a list of the product backlog for this project as a whole. The sprint backlogs are laid out in more detail in the project chapter or the sprint reports of this document. The backlogs for this project were tracked using Trello a web interface for project management. During the development process both the team and the client will have access to the board to view the progress of the current iteration of the project. During development the project will consist of six sprints of roughly three weeks each with a week after each for review as mentioned above. The projects source control is contained within a Github repository provide by the South Dakota School of Mines and Technology.

2.4.1 Sprint 1 Backlog

- Set Up Github repository

- Conduct Programming Language Research and Analysis

- Conduct Database System and Infrastructure Research and Analysis

- Conduct GUI Interface and Framework Research

- Sprint 1 Research and Sprint Report and Decision

- Begin Practicing and Learning Development Materials

- Prepare Client Presentation 1

2.4.2 Sprint 2 Backlog

- Create Starting Database Tables and Infrastructure

- Create GUI Interface Theme and Design

- Create Log In Page

- Create Permission System

- Create Landing Pages for Admin and Teachers

- Sprint 2 Report and Analysis

2.4.3 Sprint 3 Backlog

- Create Student Search

- Add a Class

- Produce a Class Role Sheet

- Create Employee Search

- Create Class Search

- Add Advanced Search to Searches

- Add and Remove Students From a Class

- Modify Student Information

- Assign Teacher to a Class

- Sprint 3 Report and Analysis

- Turn In Semester One Documentation

- Prepare Client Presentation 2

2.4.4 Sprint 3.5 Backlog

Most of Sprint 3.5 was bug fixes and putting function together in the interface. After this sprint it became clear to the team that the team would not be able to complete the student interface during this first iteration of the project. As such the student interface and the considering functionally where removed from the list of requirements upon discussions with the client.

- Role Sheet Redesign

- Tie the individual Functions Together Into Single Interface and Prototype

- Fix Various Bugs

- Add In Extra Functions Implied By User Stories

- Adapt Database After First Semester

2.4.5   Sprint 4 Backlog

- Enter Staff Pay Rates

- Enter and Update Tuition Rates

- Apply and Update Credits to a Student

- Give Early Registration Discounts

- Billing/Payment History for a Student

- Enter a Full Payment for Several Students

- Full Payment for One Student

- Allow for Payments From Multiple Sources

- Look at What A Student Still Owes

- View Teaching History

- Sprint 4 Report and Analysis

2.4.6   Sprint 5 Backlog

- Enter in Student Registration Information for Existing Students

- Modify Admin and Teacher Crossover

- Ignore Address Case to Forms

- Modify Add/Remove Students for Client Update

- Allow for and Add Multiple Source of Pay to Adapt to Client Request

- System Admin List

- Enter Staff Hours

- Add and Remove Class Location

- Remove Admin

- Remove Class

- Remove Teacher

- Remove Student

- Prorated Refunds

- Update Tuition Rates Request By Client

- Order List Functions

- Password and Username Reset Functions

- Quality Updates

- Client and Teacher Project Reformation Meeting

- Sprint 5 Report and Analysis

2.4.7   Sprint 5 Fixed Bug Backlog

- Add Remove From Teacher-Class Table to Remove Teacher Function
- Fixed Searches Errors
- Fixed Advanced Search Issues
- Remove Open Buttons
- Add More Return Buttons for Quality
- Added Dynamic Teacher and Student Schedule
- Modified Format of Functions to Better Match Database Options
- Remove Extra Navigation Bar
- Remove and Update Extra System Buttons
- Error Check Bug Fixes
- Fixed Print Functionality After Change
- Quality Updates

2.4.8   Sprint 6 Backlog

- Change Log In and Highlighted Button
- Compute Instructor Wages
- Add Student and New Student Registration
- Update to Refund Page
- Assign Discounts (if possible before submission)
- Modify Location
- Remove Command Prompt
- Final Mass Test for This Iteration of Project
- Design Fair Materials and Presentation
- Reformatting and re Factoring of Design Docs for Iteration Model
- Finish as much material for next Iteration as Possible
- Client and Teacher Project Evaluation Meeting
- Sprint 6 Report and Analysis
- Submission of Iteration Materials

2.5   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, however as pages are functional pages are shown to, or approved by the client. With the main goal of this team being to develop as much of the project as possible to hand over to the client for either the next iteration or other course of action, decided by the client.

3

Project Overview

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

3.1 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 had mostly taken on the role of scrum master within the group. However his primary role is still development of DanceSoft.

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]

3.2 Project Management Approach

The client requires a weekly meeting every Wednesday at 2:00 p.m. to check on the progress of the system during the fall 2015 semester at South Dakota School of Mines and Technology. During the spring 2016 semester the team meeting changed to Tuesdays at 2:00 p.m. and Thursdays at 3:00 p.m. if necessary. 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 for check in purposes.

The internal team management was mostly was mostly done by Marcus Berger and Dr. McGough. Marcus used the free service Trello to manage the tasks that the team needed to complete. As the project progressed the team had to reevaluate the management approach as the DanceSofts project requirements changed. After sprint 3 it was decided that due to time left for the project that the management of the student interface would need to be finish in a future iteration.

3.3   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 and artistic director of the Academy. Other academy members while not directly related also share a stock in the project development as the software directly processes and modify data given to the academy. Students and teachers will not have the access to direct evaluation of the various iterations of the DanceSoft project. However as the project nears completion in future senior design classes or however the clients choose to proceed, these group could be effected by the uses of and updates to the software. Therefore they have a secondary stake in the projects various iterations.

3.3.1   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 while providing more details on product backlog materials during the weekly meeting with the development team.

3.3.2   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. He also acts as the assessment for project progress. This means that as the project progressed Dr. McGough is able to reevaluate and reestablish requirements as the project needed or was requested by the client.

3.3.3   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 only two members, all development roles are shared between the two team members. The teams job is to develop, test, fix and adapt the various request and requirements given by the client/scrum master Dr. McGough. As developers the team takes the user stories and develops a backlog for each sprint. Then the teams job is to take these user stories and produce the code needed to complete each backlog entry.

After the backlog entry has been complete it is the developers job to test and error check the functions to make sure that the function completes all its needed tasks, in the way the team wants. Also testing is done to make sure the files don't create any problems with the rest of the software. Testing is covered in more detail in the testing section of this document.

3.4   Budget

There was no budget or monetary compensation for this project. When the project was initially laid out there was an idea for a budget that would be used for a Linux box. The Linux box would have been used to store the system on a local device. However as the requirement were brought down and redefined to an iteration of the project, rather then a full complete project, it became clear that the project would not reach a deployment state where a local machine was necessary. However the client could provided a Linux box for project storage at the end of the semester, which will not be not claimed as a budget by this project. As work continues on this project the budget may grow. Most expenditures associated with the Senior Design class an example being the DanceSoft senior design poster were covered by the DanceSoft team members. Monetary compensation for this project follow the guidelines laid out in the DanceSoft Software Contract. The team will be given and accept no form of compensation for the work on the DanceSoft or any related project. This is in accordance with the conditions of the project, client, and the senior design class.

3.5   Intellectual Property and Licensing

The intellectual property for this project is currently the property of the client Jeff McGough. However due to Dr. McGough's involvement with the South Dakota of Mines and Technology, and the senior design class, the South Dakota Board of Regents policies must be considered. These policies require that any work done by a teacher or for a teacher by a group of one or more students be submitted as property of the South Dakota Board of Regents.

The only way to avoid this policy and prevent the Academy software from being owned by the state is to make the project open source in accordance with the board's policy. As a result of this set of policies all code worked on by the DanceSoft team is stored in a public Github repository provided by SDSMT for the senior design class.

Another intellectual property and licensing policy that must be considered as part of the teams choice to use PyQt is the GPL (general public license), under which the PyQt software falls. This license requires that the user release all source code for a project using any GPL licensed code in their software. However this is only required if the client plans on distributing the software they have created. As such this license will not cause any issues for Dr. McGough or the Academy of Dance Arts since the client does not plan on distributing the code to anyone other then themselves.

Based on the goals of the project the team has constructed as software that does not violate any intellectual property rules or regulations. Therefore the current rights to the software belong solely to the client until such a time as the client choices to distribute the project for monetary gain.

3.6   Sprint Overview

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

3.6.1   Sprint 1: Initial User Story and Requirements Gathering

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

Also tackled during this phase was the research conducted into the tools the team were going to use for the project. There were three main areas the team needed to tackle to decide the frameworks for the project. First the team needed to pick the programming language that would be used for the project. Two main languages were analyzed, Python and Swift. Swift is the Mac native language recently released by Apple Inc. the benefits found for Swift were its nativity to the Mac OSX and it would be the easiest to integrate into Mac. However since Swift was relatively new the team would have to take even more time to learn the new language, and since the client did not know Swift it would be harder to analyze and make adaptions to in the future. Python had the advantage of being a language that is more universally known, both to the team, the client, and the online community that the team could turn to for support or questions. Due to this and a few other factors laid out in more detail in **Sprint Report 1** the team choose Python as the programming language for the projects primary programming language.

The other two areas of research were database and interface framework. The team looked to PyQt, Kivi, Tkinter, and other graphical interface frameworks. The team decided after looking at each one that due to the ease of use and some companies listing GUI experience, that the team would use PyQt as the framework for the interface. Then the DanceSoft team looked into database software. After looking at both SQL and non-SQL datebase options, such as MySQL and Mongo the team went with a MySQL database for the storage. As with programming language, the research results are laid out in **Sprint Report 1**

3.6.2   Sprint 2: Database Creation and Starting Pages

During this phase the database schema was constructed and implemented, being sure to keep the schema as concise as possible. The goal was to generate a database that could effectively support the needed functionality and GUI connectivity. After the database was constructed, the team moved on to the starting landing pages which are jumping off pages for functionality creation. These page are modified as phases progress and further functionality was added.

During this phase the team also plotted out the first drafts of the overall user interface flow for the desktop GUI. The team later modified this as the projects requirements were flushed out. The team would also reexamine this structure in later sprints as certain requirements were dropped, added or modified. Once the first verison of the database was up and running within the School of Mines' MySQL sever the team began working on the log in page, and the main navigation pages for both Academy admins and teachers.

The log in page is used to confirm the user of the system and check whether or not the user has admin permissions. This system is required because of the different features the admins can access within the system, such as changing pay rates or adding students or teachers to the system. The log in page then directs to the landing page selection where the user, if they have permission, can access the admin or teacher interfaces.

The first drafts of the admin and teacher interfaces were implemented in this phase as well. The admin interface contained buttons through which the user can access the various functionality of the interface, These include "Manage Employees" which takes the user to most of the employee related features within the project, also the "Manage Classes", "Manage Students", and "Manage Billing and Payroll" buttons take the user to the class, student, and billing and payroll features respectfully. The teacher interface like the admin interface contains buttons for managing students and classes, however these features differ slightly between interfaces since admins have more system access. The teacher interface also contains a "Personal Information" button which allows the user to modify certain aspects of their information in the system, such as their hours, or their user-name and password.

3.6.3 Sprint 3: Functionality Creation

This phase will compose 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.

During this phase several pages were constructed. The first of these pages was the "Add a Class" page, this pages creates a form that the user can type the information for the class in and create an entry in the database for that class.

The second page made was the add student page which allowed the user set a students registration status for specific classes to approved, or rejected from pending. This function was later pulled when the student interface was dropped from the requirement. However at the request of the client the function was left in for future adaptations and uses.

Another set of pages during this phase was the search pages. There are three types of search pages within the project, student, class, and teacher. The user is able to search any of these elements by name with the default search bar. Users can also search by different fields if they use the advanced search functionality. Lastly the users can select specific data fields and pull up all that entry's data and update it if needed.

Assign teacher allows users to select a class, if that class is assigned to a teacher already a dialog pops up asking the user if they want to reassign the class. If the class is not assigned or if the user selected to reassign the class then the function generates a list of available teacher for the given class time. The user can then select a teacher and assign them.

The log in page allows the system to deal with different user levels, and provide navigation to the different landing pages. The role sheet allows the currently signed in teacher to produce and print the role sheets for each of their classes. The last main function created during this phase was the update teacher function which allow the admins to select a teacher then a form is population with the selected teacher's information. The user can then update the teacher's information within the system. The final step of this phase was the third client presentation, which capped of the first semester and the senior design I class.

3.6.4 Sprint 4: Development Updates and Payroll/Billing

This phase consisted of updates to phase three functions, adding new functions, and adding in the payroll function to the project. The first part of this phase was sprint 3.5, during this time the team added several bug fixes such as assign class and connecting functions together for a connected prototype.

One of the pages completed during this sprint was the teaching history function. This function displays all the classes the teacher has taught in the recent time frame. Another page tackled was the student billing invoice in this function the amount owed and the amount the student has payed is displayed along with

Figure 3.1: Add Class Form

the classes the student took to generate that amount. Alongside this function the team created the ability to process a payment for a student. The user is allowed to enter in the amount the student paid and the payment type. The user can also process a full payment if the student elected to pay the full amount.

As a payment is processed it is added to the student billing history which can then be viewed and printed by the user. The billing history contains the payment id, the name of the student, the amount paid, and the date of payment. Another function implemented during this phase was the students credit interface, within this page the user can select a student and then apply a credit to that student. The credits do not directly connect to any other function due to the fact that the Academy handles credits on a student by student basis at the discretion of the Academy owner Julie McFarland.

Similar to credits the system also includes the functionality for the modify tuition and fees within the database. The tuition page allows the user to select the tuition name and change the rate. The user can also hit the add button on the page which opens a dialog box where they can put in the required information. Other buttons included within this page are update and delete which allow the user to update an existing entry a=or delete an entry respectfully . The fees page has the same structure as the tuition pages, the user is able to select a fee and update or delete the fee rates. There also is the ability when add or updating a fee to mark it as a percent so the value given must be a decimal from zero to one. The last function added during this phase was the ability to enter in teacher pay rates, this function allows the admin in the system to declare a pay name and pay rate that a teacher has and the amount of hours they worked. The function then calculates the gross wage of the teacher based on those pay rates. The teacher side also has an enter hours pages which allows the user to enter in the amount of hours they worked under each pay type.

3.6.5   Sprint 5: Updates Bug Fixes and Functionality

During sprint five the team tackled the remain functionality, the sprint rollover and various bug fixes. The first of these fixes was the ability to update an admin or teachers information and have the updates crossover between the two tables in the database. The address in the forms page where also modified to always be uppercase to avoid the occurrence of two similar names within the database. Another modification made was to the approve/reject student page, when the page was initially created the team did not allow rejected students to be re approved, this function was then adapted to allow for this to be closer to the client's request. Change user credentials were also added during this phase.

An admin list was added that displays the admins for the system and allows the addition and removal of admins from the system. Alongside this function several removal functions where added. These include: remove student, remove class, remove class location, and remove teacher, the function are final purges for the data. This means that if a user removes a teacher the teacher information, classes taught, classes assigned, and account information are all removed from the system. For students the delete also includes all the billing and student data, so the system delete functions should only be used when the user is absolutely sure that the information is not longer needed.

The main function completed during this phase was the student registration interface. This allows the Academy to enter in new students to the system, and update the student. The user can then pull up a list of available classes and classes the student is regesteer for and add them to classes. These classes are them added to the student-class table and the classes are processed by the system. The user can then view and print out student and or teacher schedules.

Bug Fixes Completed During Phases 4, 5, and 6:

1. Date time update bug

2. Form information bug

3. Add teacher class to remove teacher

4. Spelling errors

5. Remove class cost

6. Search update bug

7. Search refresh bug

8. Advanced search bug

9. Error checks

10. Remove uneccicayry button

11. Dynamic times on schedules

12. Few combo box changes

13. Text changes

3.6.6   Sprint 6: Updates, Fixes, and Iteration Packaging

During the last phase the team attempted to finish as much as possible before turning in this iteration of the project. This phase consisted mostly of bug fixes and client updates after demoing the final project the team managed to create. First since the remove student showed all students in the system the team added a search bar to clean up the functions execution as much as possible. Second entering teacher hours was refined a bit to allow for a default course hours rate. Next several bugs were fixed, and modify location functionality was added.

Student registration was also complete during this phase. The last half of this phase was focus on prep and execution of the design fair presentation held by the South Dakota School of Mines and Technology. Overall this last phase consisted of the last step of senior design and preparing to hand off our teams iteration of the DanceSoft project.

3.7   Terminology and Acronyms

1. Backlog - A list of task to be completed

2. Budget - money provided by the client to supply the needed materials for the project

3. Database - A storage and platform to manipulate data for the projects

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

5. Sprint - three week time periods where portions of the project are completed

6. Timeline - the plan of when a assignment is to be completed

7. GPL - General Public License - a type of software development license

8. SDSMT - the South Dakota School of Mines and Technology's university acronym

9. SDBOR - South Dakota Board of Regents

10. Interface - The screen or set of screens that the user can see and navigate within the project

11. Source Control - A system used to manage multiple people working on the same collections of infromation to maintain consistency,

3.8   Sprint Schedule

There are three sprints for Fall semester and three for Spring semester

1. Sprint 1: 9/14/15 - 10/2/15

2. Review and Client Presentation: 10/20/15

3. Sprint 2: 10/12/15 - 10/30/15

4. Sprint 3: 11/9/15 - 11/27/15

5. Review and Client Presentation: 12/3/15

6. Sprint 4: 1/18/15 - 2/5/16

7. Sprint 5: 2/15/16 - 3/4/16

8. Review and Client Presentation: 3/22/16

9. Sprint 6: 3/21/16 - 4/15/16

10. Design Fair: 4/19/16

3.9   Backlogs

3.9.1   Sprint 1 Backlog

- Set Up Github repository
- Conduct Programming Language Research and Analysis
- Conduct Database System and Infrastructure Research and Analysis
- Conduct GUI Interface and Framework Research
- Sprint 1 Research and Sprint Report and Decision
- Begin Practicing and Learning Development Materials
- Prepare Client Presentation 1

3.9.2   Sprint 2 Backlog

- Create Starting Database Tables and Infrastructure
- Create GUI Interface Theme and Design
- Create Log In Page
- Create Permission System
- Create Landing Pages for Admin and Teachers
- Sprint 2 Report and Analysis

3.9.3   Sprint 3 Backlog

- Create Student Search
- Add a Class
- Produce a Class Role Sheet
- Create Employee Search
- Create Class Search
- Add Advanced Search to Searches
- Add and Remove Students From a Class
- Modify Student Information
- Assign Teacher to a Class
- Sprint 3 Report and Analysis
- Turn In Semester One Documentation
- Prepare Client Presentation 2

3.9.4 Sprint 3.5 Backlog

Most of Sprint 3.5 was bug fixes and putting function together in the interface. After this sprint it became clear to the team that the team would not be able to complete the student interface during this first iteration of the project. As such the student interface and the considering functionally where removed from the list of requirements upon discussions with the client.

- Role Sheet Redesign

- Tie the individual Functions Together Into Single Interface and Prototype

- Fix Various Bugs

- Add In Extra Functions Implied By User Stories

- Adapt Database After First Semester

3.9.5 Sprint 4 Backlog

- Enter Staff Pay Rates

- Enter and Update Tuition Rates

- Apply and Update Credits to a Student

- Give Early Registration Discounts

- Billing/Payment History for a Student

- Enter a Full Payment for Several Students

- Full Payment for One Student

- Allow for Payments From Multiple Sources

- Look at What A Student Still Owes

- View Teaching History

- Sprint 4 Report and Analysis

3.9.6 Sprint 5 Backlog

- Enter in Student Registration Information for Existing Students

- Modify Admin and Teacher Crossover

- Ignore Address Case to Forms

- Modify Add/Remove Students for Client Update

- Allow for and Add Multiple Source of Pay to Adapt to Client Request

- System Admin List

- Enter Staff Hours

- Add and Remove Class Location

- Remove Admin

- Remove Class

- Remove Teacher

- Remove Student
- Prorated Refunds
- Update Tuition Rates Request By Client
- Order List Functions
- Password and User Name Reset Functions
- Quality Updates
- Client and Teacher Project Reformation Meeting
- Sprint 5 Report and Analysis

3.9.7   Sprint 5 Fixed Bug Backlog

- Add Remove From Teacher-Class Table to Remove Teacher Function
- Fixed Searches Errors
- Fixed Advanced Search Issues
- Remove Open Buttons
- Add More Return Buttons for Quality
- Added Dynamic Teacher and Student Schedule
- Modified Format of Functions to Better Match Database Options
- Remove Extra Navigation Bar
- Remove and Update Extra System Buttons
- Error Check Bug Fixes
- Fixed Print Functionality After Change
- Quality Updates

3.9.8   Sprint 6 Backlog

- Change Log In and Highlighted Button
- Compute Instructor Wages
- Add Student and New Student Registration
- Update to Refund Page
- Assign Discounts (if possible before submission)
- Modify Location
- Remove Command Prompt
- Final Mass Test for This Iteration of Project
- Design Fair Materials and Presentation
- Reformatting and re Factoring of Design Docs for Iteration Model
- Finish as much material for next Iteration as Possible
- Client and Teacher Project Evaluation Meeting
- Sprint 6 Report and Analysis
- Submission of Iteration Materials

3.10 Development Environment

The basic purpose for this section is to give a developer all of the necessary information to setup their development environment to run, test, and/or continue development of the DanceSoft project, while also providing information on the systems the team used to create the project.

3.10.1 Source Code Development

The source code for this project was written in Python. Due to the cross platform nature of this language the team used a variety of different IDEs over the course of development, these included:

1. Python IDLE

2. Microsoft Visual Studio 2015

3. Xcode

Any developers wishing to continue work on this project should be able to run this software within any Python 3 capable interface of their choice.

3.10.2 MySQL Database

The projects back end contains a MySQL database. MySQL can be installed on a developers computer free of charge off of the MySQL website. This package also contains a helpful GUI interface called MySQL workbench which can aid the user in manipulating the database if they so choose. However the system should be editable within any MySQL enabled database manipulation software. Once a developer has MySQL installed they simply need to log in and connect to the database using credentials provided by the client.

3.10.3 PyQt

The projects front end interface is developed using PyQt. PyQt is an extention of Qt it allows the development of Qt GUI's and functionality in Python. The tool kit contains all the normal Qt widgets including: line edits, spin boxes, combo boxes, text edits, and other normal GUI widgets. PyQt also has a designer that the team used to create the major interface pages. The designer allows the user to click and drag components and develop the pages in a more visual environment. The designer can then create generated code for the pages using the pyuic4 command in a terminal as follows, "pyuic4 -o generatedCode.py uiFilename.ui" this command produces a .py file containing the generated designer code. The generated class can then be included in the python files to use the code. In classes written by a developer, programs can include any of the major Python 3 or PyQt libraries.

Some common libraries the team used where:

- QtGui

- QtCore

- PyQt4.QtSql

3.11 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 compatibility 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 further 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/`

3.12 Source Control

Source control for this project was conducted using Github, and the Github GUI. Github is a git add in, a developer could use whatever git manager they want. The Github GUI can be installed from www.github.com. The repository was provided by South Dakota School of Mines as part of the senior design class. A developer wishing to continue work on this project simply need a git software and access as a contributor to the repository. However once the clients move the code to a local device a developer will need access to this device as well.

3.13 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.

3.14 Build Environment

A user, or developer can build the project through the use of compiled python scripts. A user can run the log in script which will compile some of the Python scripts as to speed up various aspects of running the project. There are no special build scripts that are requires before the project can be run a user simply needs to run the login script for the DanceSoft project.

3.15 Development Machine Setup

1. A machine will need to be acquired that is capable of running PyQt and MySQL

2. The user will need to install python 3 from `https://www.python.org/download/releases/3.4.3/`

3. Next the user need to install the PyQt 4 library and the Qt designer if necessary from `https://riverbankcomputing.com/software/pyqt/download`

4. After downloading Pyqt the user need to download MySql from `http://dev.mysql.com/` and connect to the database where the DanceSoft database is stored.

5. Now if the PyQt libraries have been installed correctly the user should be able to run the PyQt code and proceed to develop the DanceSoft project.

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 Architecture and System Design

This is where you will place the overall system design or the architecture. This section should be image rich. There is the old phrase *a picture is worth a thousand words*, in this class it could be worth a hundred points

(well if you sum up over the entire team). One needs to enter the design and why a particular design has been done.

4.1.1 Design Selection

Failed designs, design ideas, rejected designs here.

4.1.2 Data Structures and Algorithms

Describe the special data structures and any special algorithms.

4.1.3 Data Flow

4.1.4 Communications

4.1.5 Classes

4.1.6 UML

4.1.7 GUI

4.1.8 MVVM, etc

4.2 Major Component #1

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. Here is an example code listing:

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

```c
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.

4.3 Major Component #2

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.

4.4 Major Component #3

4.4.1 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.

4.4.2   Component Overview

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

4.4.3   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.

4.4.4   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.

4.4.5   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.

4.4.6   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 testing the DanceSoft project.

5.1   Overview

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.

Each requirement (user story component) should be tested. A review of objectives and constraints might be needed here.

5.2   Dependencies

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

5.3   Test Setup and Execution

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. One approach is to list each requirement, module, or component and describe the test.

The unit tests are described here.

5.4   System Testing

5.5   System Integration Analysis

5.6   Risk Analysis

During development of the DanceSoft project the project encountered a few possible risk that could effect the system. The team has found the following main risk within the system.

1. Interface Usability

2. Database Connectivity

3. Data Security

4. Data Backup

The first of the main risk is the PyQt interface usability. The system must maintain a simple and easy to use interface. This way the users can efficient and effectively access the system while maintaining the functionality at the systems core. The team must always keep this fact in mind when developing any of the systems pages and structures, as the users at the Academy are not assumed to have a technical background.

The second risk is the database connectivity. If the database can not be reached the project is incapable of accomplishing its functions. The system at its core is a database manipulation software, without the ability to connect to the database the project can not function. The system must therefore assure connection before allowing the users to continue.

The third set of risk with the DanceSoft system is database backups and data security. Database backups can be achieved by using functions within the MySQL database software. These backups can then be stored on a local system or wherever the client chooses to store the data. Database security can be achieved using prepared statements and queries, and other techniques of encryption and protection.

5.6.1   Risk Mitigation

The following section is the teams approaches, goals, and ideas to mitigate the various risk with this the DanceSoft project.

The teams has been developing each interface with the idea of usability in mind during the entire process. One of the testing approaches that the team used was user testing. This method meant sitting down and acting like a user of the system and not a developer or technical user. If during this process something did not make sense or execution did not work in the expected way, the interface or function was reworked. Another part of this approach involved having people do sudo-beta test by sitting and attempting navigation of the system. If any questions or concerns came up from the user they were analyzed and addressed if necessary by the team. By keeping the interface simple and asking for feedback the team hopes to mitigate usability issues as much as possible.

The database risk are only partially dealt with in the current project. The database connection issues are handled through a few error checks in the connection. Also since the database will be contained in the local machine provided by the client the database can be connected to directly. This connection issue will become greater once a student web interface is added to system, or if the client moves the database the connection will become reliance on the communication between either the two systems or the network connectivity at the Academy. Some of these future student interface issues were discussed before the student portal was dropped from the project. The rink could possibly be handled by storing the information the students are sending through a socket and storing it until the database is booted at the Academy. At which point the updates would be submitted. Another database risk is the security of the data stored within. The backup issue can be solved through writing the contents to a file which can be stored on either the local machine, or another machine as to protect the data should something happen to the computer in which the database is stored. Many MySQL management software also have ways of exporting the statements needed to create the database which the client can use to manage backups and create text file backups. The actually security of the data will not be completed in this iteration of the software. However the team has a few ideas for data security. Firstly since only the local system exist at this time, data should not be passed over a network. The SQL queries can be placed into prepared statement to aid in their security. Its is highly advised by the development team that security solutions be explored by future developers as the current team lacks the experience and expertise to truly ensure the security of the data contain within the system.

5.7   Successes, Issues and Problems

5.7.1   Changes to the Backlog

Overall the project's base remained unchanged from the the beginning requirements. However as the project was being built the backlog and project encountered a few changes and iterations. These changes and modification are briefly explained in this section.

During the project the backlog was changed multiple times. The first of these redesigns occurred after sprint 2. The team and the client elaborated on some of the user stories to give the team an increased amount of focus for the various functions. This allowed the team to better develop the core system functionality required by the project.

However the first major change to the project's backlog occurred after sprint 3.5. It became clear to the team that the student web portal would not be completed by the end of the senior design project period. After discussions with the client in order to make the local desktop GUI as complete as possible the student interface was dropped from the project requirements. Users stories for this aspect of the project included: student registration, view tuition, and student schedules. In response to these changes the team ported the functions into the local desktop interface to create the student registration, billing, schedule, and other interfaces.

The last changes to the backlog occurred in the last sprint of the project. The project had been refined at this point as a first iteration of the DanceSoft project. This change became necessary as issues arose within the team that prevented the team from reaching full project completion. As a consequence of this the project became a proof of concept or minimum viable product for the client. The team demoed the existing software for the client, at this meeting the client and the team discussed the requirements again and reevaluated the project requirements for completion, and the senior design fair presentation.

6

Prototypes

This chapter is for recording each prototype developed. It is a historical record of what you accomplished in 464/465. This should be organized according to Sprints. It should have the basic description of the sprint deliverable and what was accomplished. Screen shots, photos, captures from video, etc should be used.

6.1   Sprint 1 Prototype

6.1.1   Deliverable

6.1.2   Backlog

6.1.3   Success/Fail

6.2   Sprint 2 Prototype

6.2.1   Deliverable

6.2.2   Backlog

6.2.3   Success/Fail

6.3   Sprint 3 Prototype

6.3.1   Deliverable

6.3.2   Backlog

6.3.3   Success/Fail

6.4   Sprint 4 Prototype

6.4.1   Deliverable

6.4.2   Backlog

6.4.3   Success/Fail

6.5   Sprint 5 Prototype

6.5.1   Deliverable

6.5.2   Backlog

6.5.3   Success/Fail

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

8.1 User Guide

The following is the user guide for the DanceSoft project:

8.1.1 Entering the Software

Login page:
When the user first logs into the system the user is prompted with a log in window where they can enter their user name or pass word for the system. If the user enters the wrong password then the system pops up a dialog and tells the user to reenter there password. The default user name for a new user is the name used to enter the system. So if I enter a new teacher named Marcus Berger then the default user name is Marcus Berger. The default password is rcdance, both the user name and password can be changed using the change functions in the personal section of the teacher landing page.

Landing Selection:
After the user logs in they can select which of the two landing pages, Admin or Teacher that they want to go to. If the user does not have admin permissions the user will be blocked from going to the Admin Landing page.

8.1.2 Admin Landing

Once an admin has logged in they can select the admin landing page to be taken to a variety of functions listed below. In the main page the user has four options for type of functions to choose from employee, student, class, and billing.
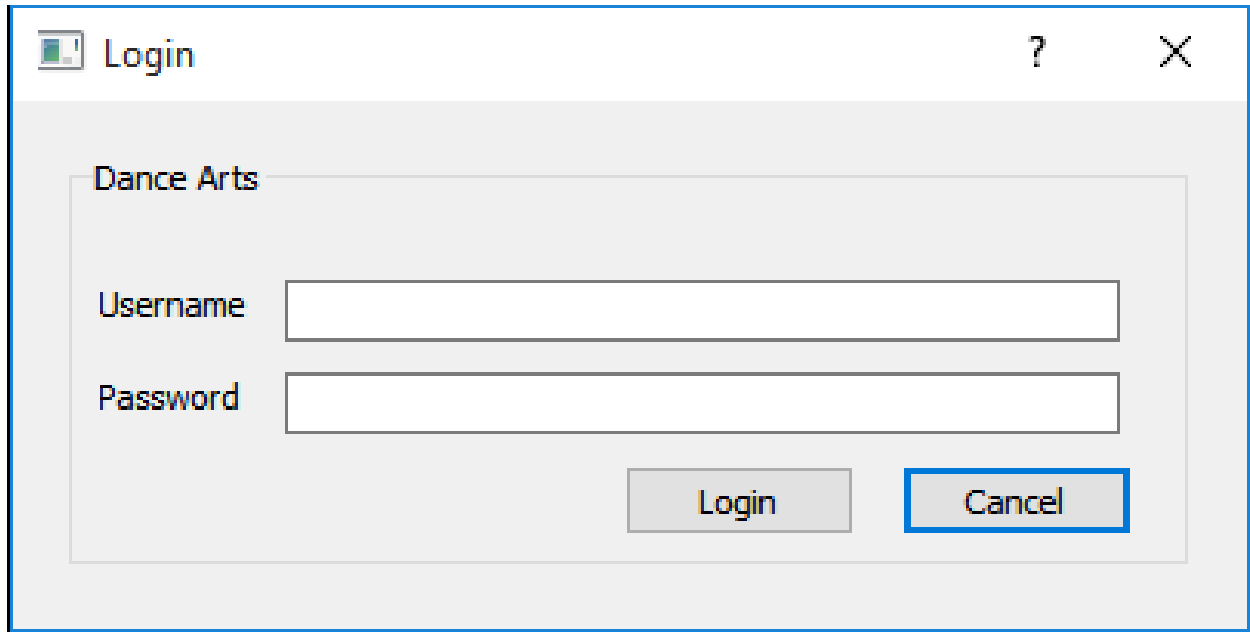
8.1.2.a Manage Employees

When the user clicks on manage employee they are given a set of buttons that they can choose from.
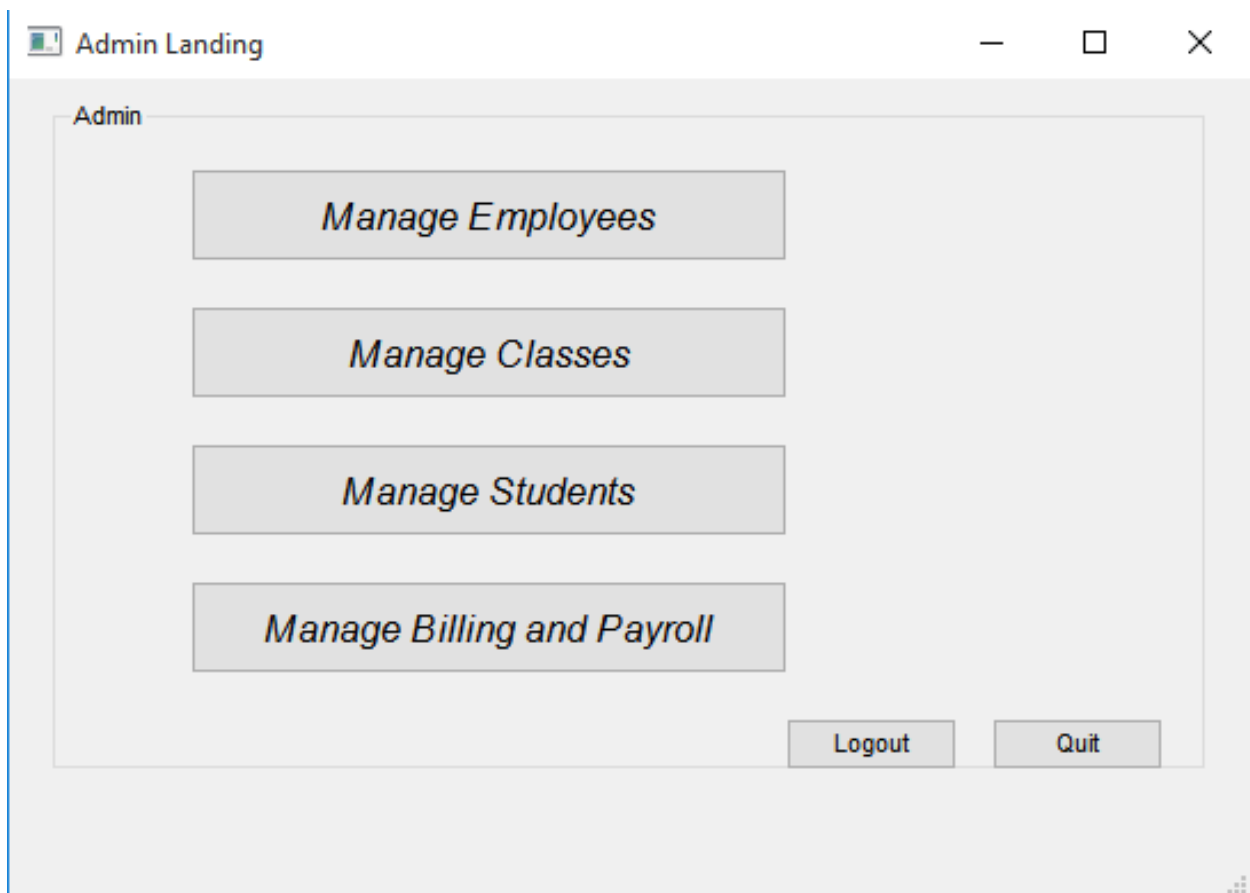
Search Teacher:
When the user selects search teacher, the system will bring up a window that displays the teachers in the system. The user can then type a teacher's name into the search bar and the field will display the teachers that names contain the entered value. If the user is looking for an exact match then they can check the "exact check box. After the user conducts a search, they can click the refresh button to bring back the full list of teachers in the database.

By clicking the advanced search button the user can further refine the search value, the advanced search dialog is accessed by clicking the advanced search button. Advanced search allows the user to to search for teachers by id, name, phone numbers, or date of birth.

Figure 8.1: Log in page

Figure 8.2: Admin Main Page

Figure 8.3: Teacher Search Window

One the users desired teacher is found the user can click on the name in the main window and select a details button to view all the information for that teacher. In the details window the user is able to change the information for an entry and submit the updates to the system.

Show Admin List
On the main admin page the user can select the admin list button. This will display a text list of admins for the system. The window contains a search bar to refine the list if the user is looking for a specific entry. The page contains a details button which will allow the user to view the full details of the entry. The add button allows the user to select a teacher in the system and give them admin permissions. The remove button allows the user to remove admin permissions from the system, this however does not remove the teacher from the system.

Update Teacher Information:
The update teacher button on the admin employee page, allows the user to select a teacher in the system and one selected a form will be populated. The user can then change any of the fields and submit the updates to the systems database.

Enter Teacher Hours:

Figure 8.4: Update Teacher Form

Figure 8.5: The Admin Hours Page

Another button on the admin landing page is "Enter Teacher Hours". The user is able to select a teacher and a dialog pops up. The user is then able to view the selected teachers hours in each pay rate and change them if the user so chooses. The gross wage is then calculated and displayed to the user.

Assign Teacher to Class:
This button bring up the assign teacher window. From here the user is able to select a class name, if the class is currently assigned to a teacher a dialog box will pop up asking the user if they would like to reassign the class to a different teacher. If the user selects yes or if the selected class is currently not assign to a teacher a list of teachers available at the class time will pop up and the user can select a teacher to assign to the class.
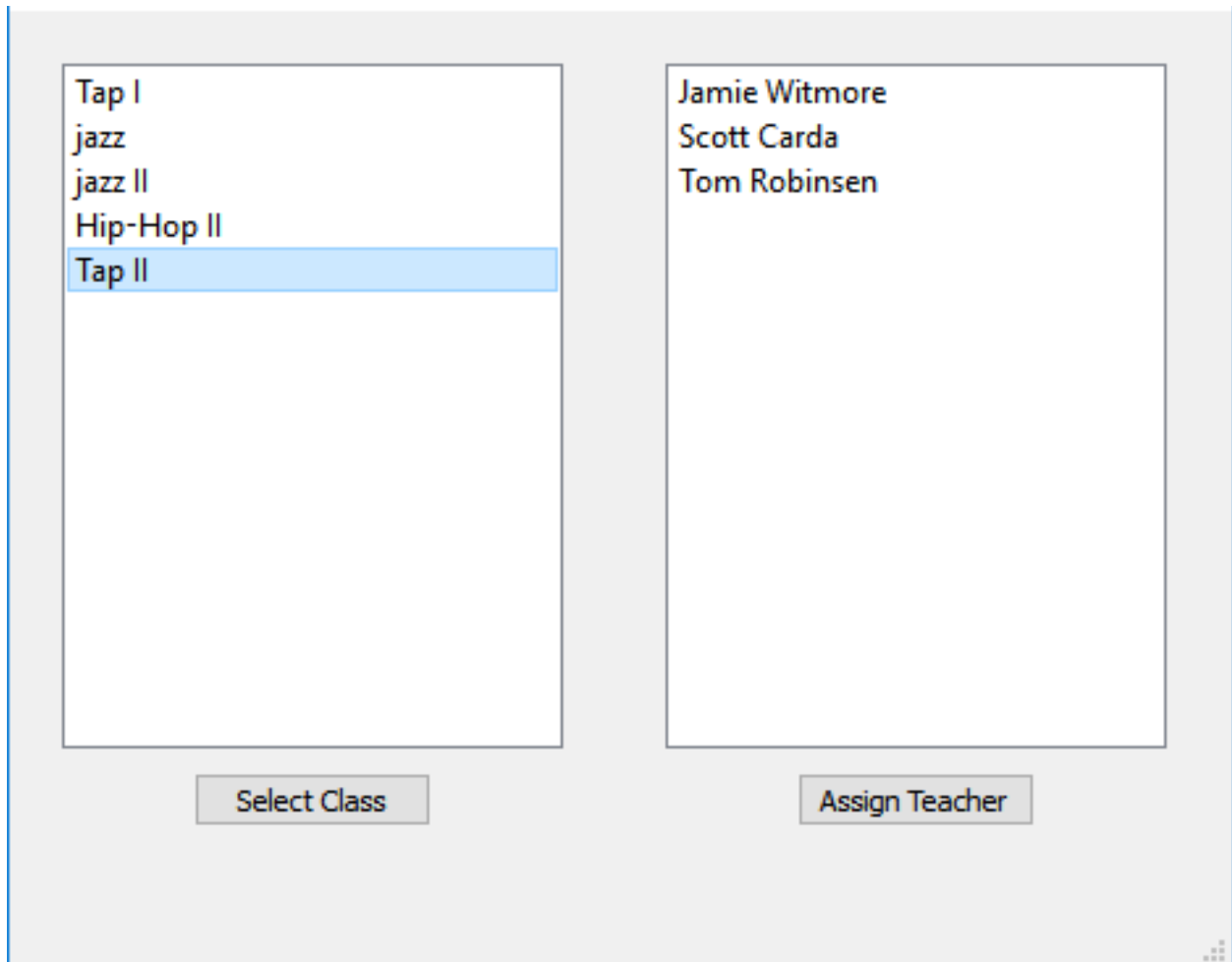
Teaching History:
The user is able to select a teacher and press the history button, this brings up a list of all the classes that teacher has taught, excluding the current semesters classes.

Enter New Teacher:
This button ups a form where the user can enter the fields for a new teacher. Once all the required fields are filled out the user can submit the teacher to the system. The required fields are name, date of birth, home phone, address fields, gender, and SSN. When the submit button is clicked, the user will be able to review the entry before submitting it to the system.

Remove Teacher
The user is able to select the name of a teacher and remove the selected teacher from the system. This removal is complete, this means that if the user removes a teacher it will completely remove all the data from the system connected to that teacher. This includes classes, teacher data, and hours.

Tap I
jazz
jazz II
Hip-Hop II
Tap II

Jamie Witmore
Scott Carda
Tom Robinsen

Select Class

Assign Teacher

Figure 8.6: Assign Class Window
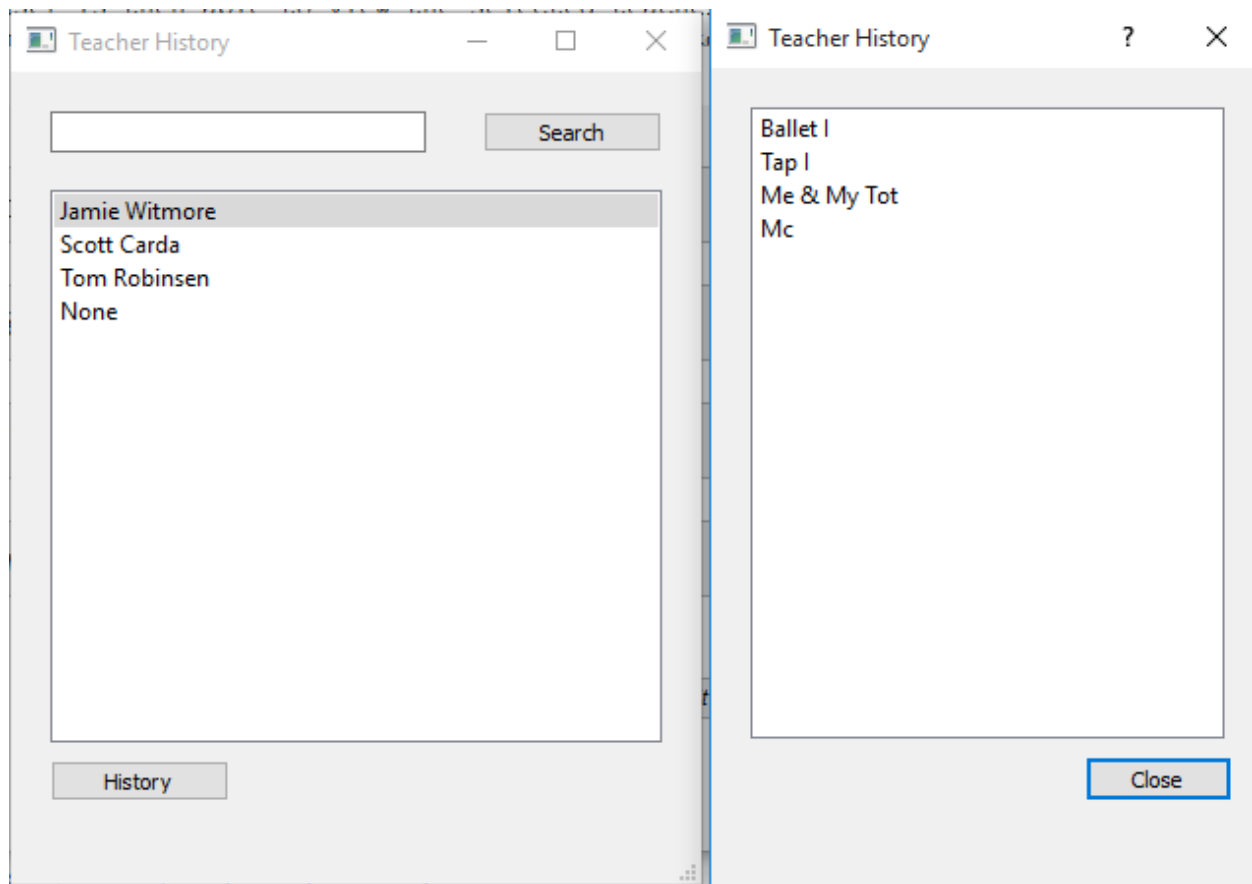
Figure 8.7: Teacher History

Figure 8.8: Class Search Window

8.1.2.b   Manage Classes

View Classes
This button allows the user to search classes like in the teacher search explained above. The general functionality is the same, the user is able to search by class name and refresh the view window after a search. The user can check the exact button if they wish to look only for what they entered. The Advanced search button allows the user to search for classes based on id, name location and time. Once the desired class is found the user can select a class and press the details button in order to view the full details for a course. Within this details page the user can modify the information and send updates to the system.

Add a Class:
A form will pop up where the user can enter in all the information needed to add a class. Once the user has entered in all the information and clicked submit a dialog will pop up where the user can check and confirm the submission. Selecting "Add New Location" in the location drop down will pop up a dialog box which will allow the user to enter in a new location for the class, the user can also select a location that already exist in the system.

Remove Class:
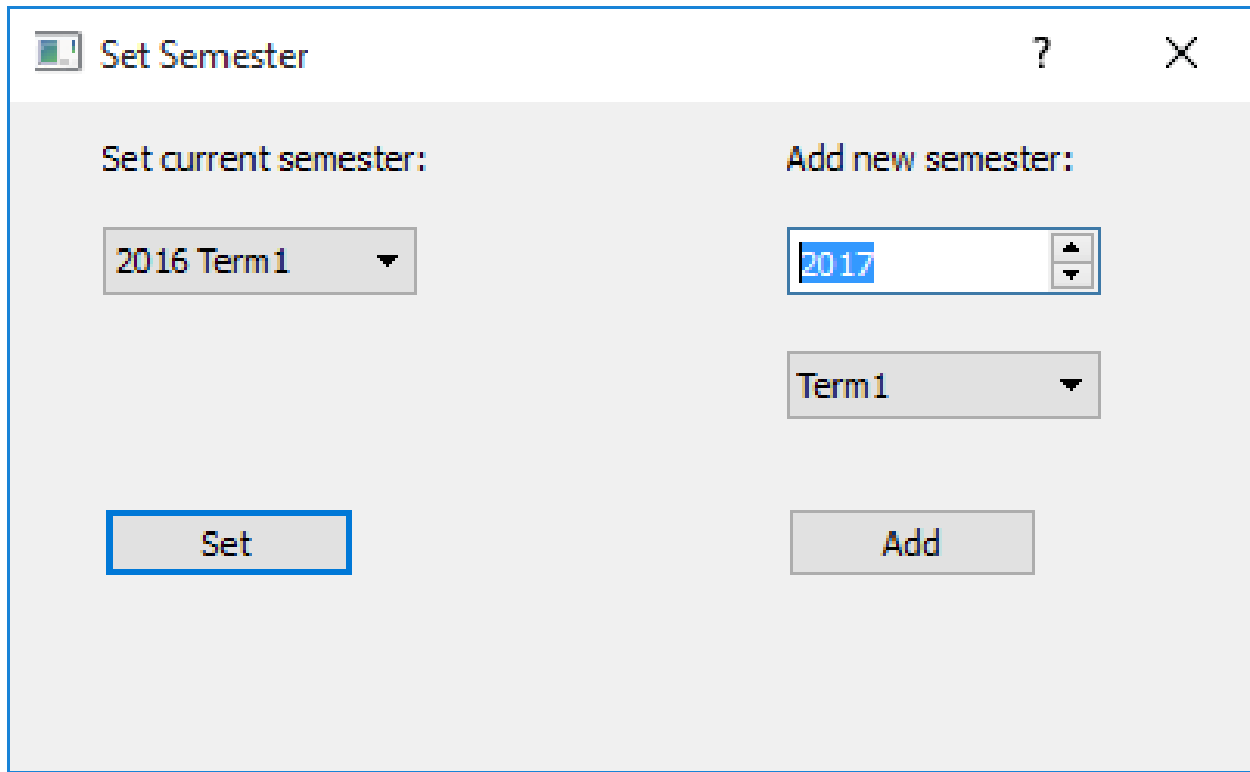
Figure 8.9: Set Semester Window

The user is able to select the name of a class and remove the selected class from the system. This removal is complete, this means that if the user removes a class it will completely remove all the data from the system connected to that class. This includes class data, removal from schedules, removal from any billing calculations, and any pending registrations

Set Semester:
Allows the user to select the current semester in the system or add a new semester.

8.1.2.c   Manage Students

Search Students:

When the user selects search student, the system will bring up a window that displays the students in the system. The user can then type a student's name into the search bar and the field will display the students that names contain the entered value. If the user is looking for an exact match then they can check the "exact check box. After the user conducts a search, they can click the refresh button to bring back the full list of students in the database.

By clicking the advanced search button the user can further refine the search value, the advanced search dialog is accessed by clicking the advanced search button. Advanced search allows the user to to search for students by id, name, phone, guardian or date of birth.

One the users desired student is found the user can click on the name in the main window and click a details button to view all the information for that student. In the details window the user is able to change the information for an entry and submit the updates to the system.

Student Credits:
This button will pull up a window where the user can select a students name and their credit amount within

Figure 8.10: Student Search Window

Figure 8.11: Student Credit Window

the system will be displayed. The user can then update the credit amount for that student. The credits are not factored into any student billing calculations. This is in accordance with the Academy of Dances Arts refund and credit policy as the credits and refund distribution is up to the users of the system. One the credits have been used the user must reset the reset the students credits to zero.

Add/Remove Student:
This button brings up a window that allows the user to select a class. Once a class is selected the students currently registered will appear al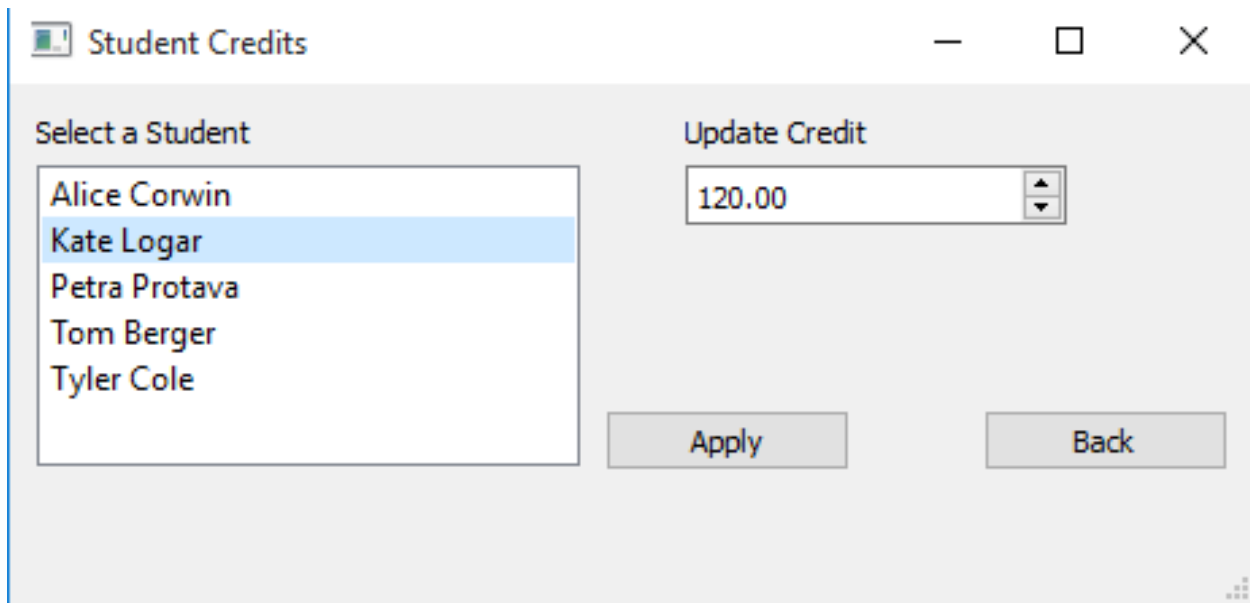ong with their class approval status. User can then select a student name and either click the add button to set the approval to one meaning the student is approved to take the class. A user can also remove the student which sets the approval to negative one meaning rejected. An approval of zero means the students registration is pending.

This function was created for the student interface links, however this set of functions were dropped so this function was left in at the request of the project's initial client Dr. Jeff McGough, so a future iteration of the project could see it. However in the projects current state this window serves no real overall purpose and should not be used by any users of the software. Anyone wishing to do student registration should do so through the student registration function explained below.

Registration:
The user is able search for a preexisting and select a student. The user can then click the "Update" button to bring up the details of that student, the user can then change and update the student's information in the system. Once the updates are made or if there are no information updates the user can click the "list" button and bring up a list of classes the student is currently registered for and the classes the student can register for. Classes can be dropped by selecting the class and clicking the drop button. Classes can be added from the pending class section using the add button.

Back on the student list window instead of clicking the "Update" button, the user can click the "Add" Button and enter in a new students information. If the new students guardian is not in the system the user can click "Add Primary" to add the student's primary guardian. If the student needs a new secondary guardian the user can click the "Add Secondary" button and add another guardian to the system. Once all the information for the student has been added the user can click the "Add" button to finalize the student.

Remove Student:
The user is able to select the name of a student using a search bar and remove the selected student from the system. This removal is complete, this means that if the user removes a student it will completely remove

Figure 8.12: First Student Registration

Figure 8.13: Current Student Registration Window

Figure 8.14: Add New Student Window

Figure 8.15: Partial Payment Window

all the data from the system connected to that student. This includes class data, removal from schedules, removal from any billing calculations, any pending registrations, and if no other students such as sibling are connected to an address or a guardian the address and/or guardians will be removed as well. Any remove functions should be done with caution

8.1.2.d   Manage Billing

These functions are the general billing function for this version of the software ane allow the user to handle the base level billing functionality of the DanceSoft project.

Enter Partial Payment:
The User is able to search for a student and select them. When the user selects a student that is making a payment a window will appear where the user can type in the amount paid by the student. The type of payment is selected from a drop down menu, this value is used in the billing history to display how the student paid. The last combo box allows the user to select the term in which the payment was made. The payment is processed when the user clicks "Confirm."
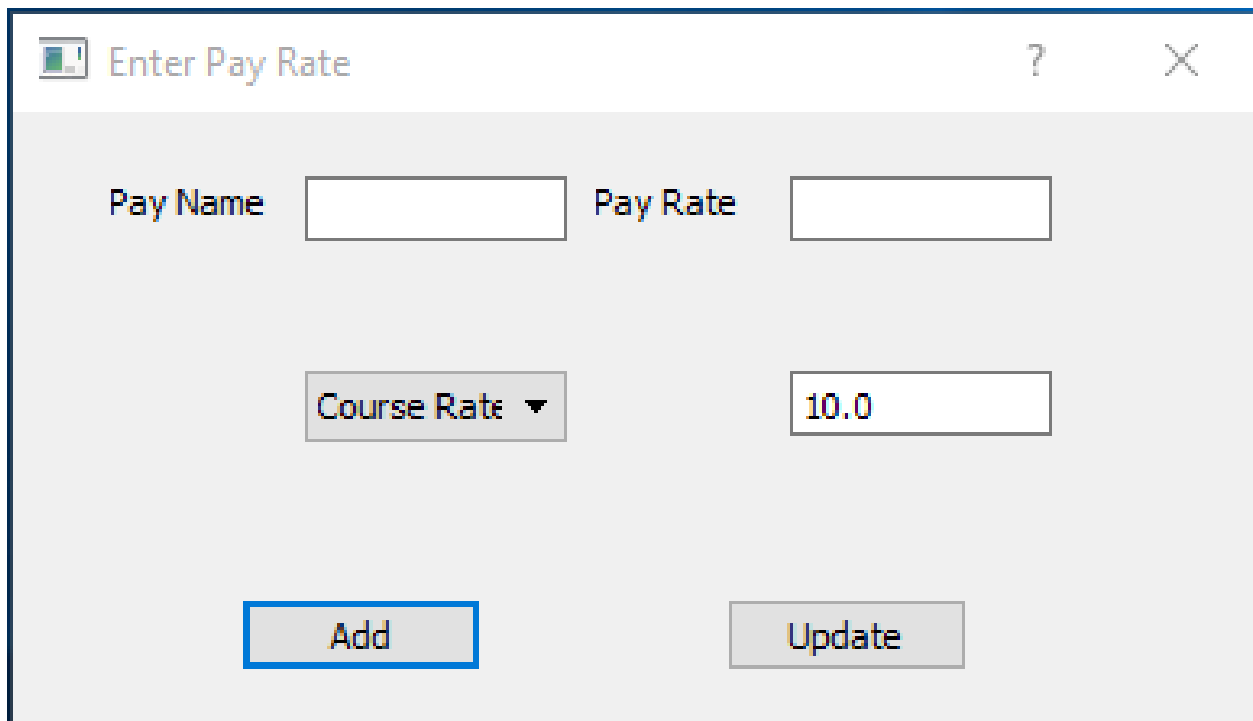
Enter Full Payment:
Enter full payment works the same way as partial payment the user is able to search for an select a student, however instead of entering a payment by clicking the button the students owed amount will be set to zero.

Enter Teacher Pay Rate:
The Enter Teacher Pay Rate button opens up a window where the user can select a teacher. Once a teacher is selected the user can open the pay rate window. From the pay rate window the user can enter a new pay rate for the current teacher and the amount that pay rate is worth. The user can them add the pay rate to the system by clicking the "Add" button. If the user chooses to update the pay rate the user can select the

Figure 8.16: Enter Pay Rate Window

rate they wish to update from the drop down menu and change the value in the box. The update is submitted by clicking the update button. Currently the added pay rates are for that teacher, but the course rate is the same among all teachers. Future versions should add the ability to have different course rates for each teacher.

Student Balance:
The user selects a student and can clicks the "Statement" button. The students statement will produce a name, the current semester, the classes taken that semester, the total money paid, due, and the remaining balance. The user can print out the statement as well if needed.

Billing History:
When the user clicks the "Billing History" button, a window appears where the users can search for and select a student. Once a student is selected the user can click the statement button and produce the payment list for a student. The payment list contains the name of the student, the term the payment was made for, the amount paid, and the date the payment was made. The user can print the statement if they want to using the print button.

Tuition Rates and Fees:
The user is able to view all the tuition rates in the system and update them if need be, by changing the values of in the boxes. The user can also add and remove tuition rates as well. In the fee window the user can add a fee update a fee in the system and remove a fee it they so choose. However as of this version of the project none of the fees are factored in to any of the calculations. In future versions the project should be able to connect the fees to billing calculations. The tuition rates are based on the amount of minutes and they in turn determine the amount a student owes. If these are changed or remove it could mess with the way the system does billing for the Academy. Handle with caution. In future iterations teams should be able to better connect these systems.

8.1.3 Teacher Landing
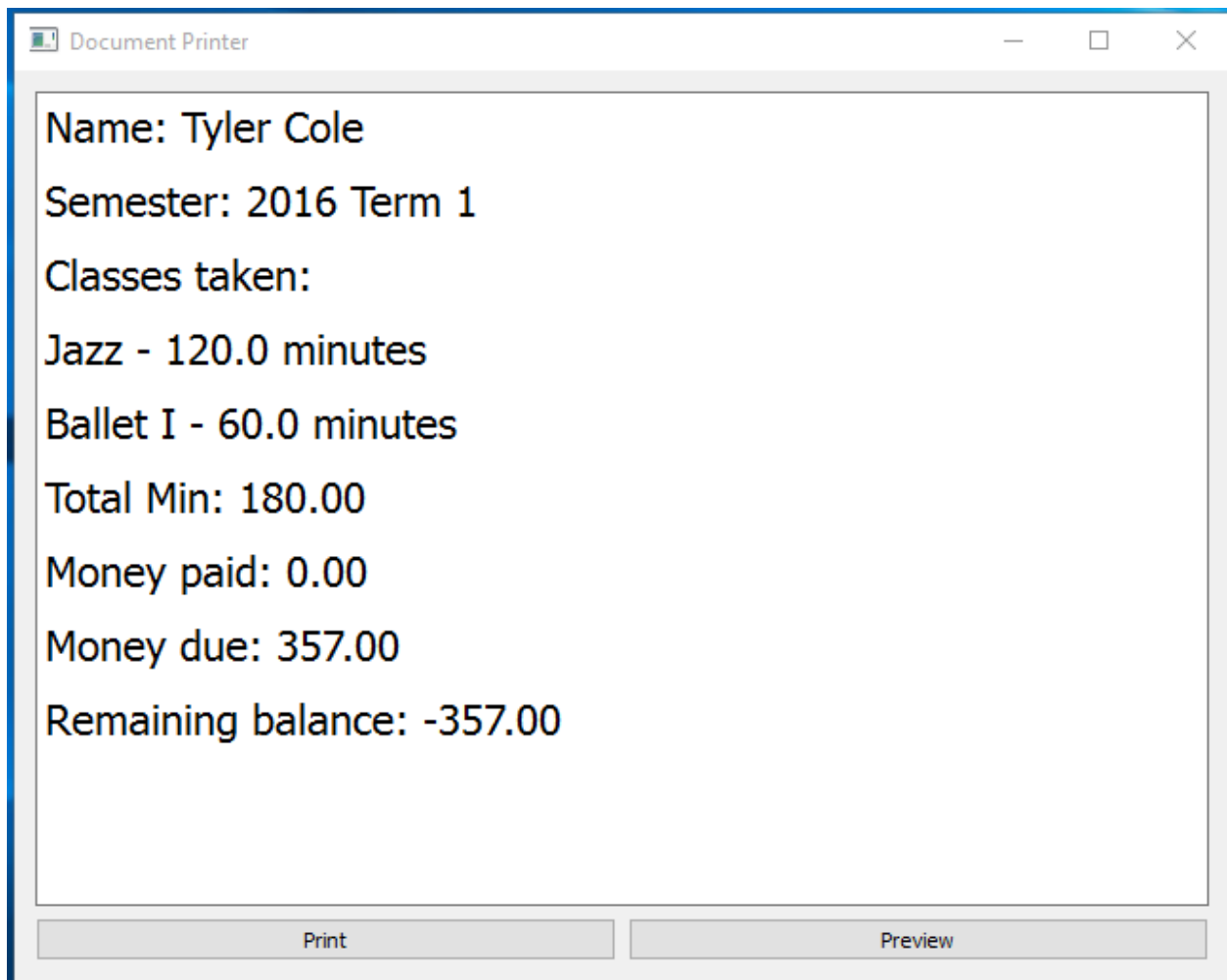
Figure 8.17: Student Balance Window

Figure 8.18: Student Billing History

Figure 8.19: Tuition and Fees

Figure 8.20: Student Search Window

8.1.3.a   Student Options

Search Students:
When the user selects search student, the system will bring up a window that displays the students in the system. The user can then type a student's name into the search bar and the field will display the students that names contain the entered value. If the user is looking for an exact match then they can check the "exact check box. After the user conducts a search, they can click the refresh button to bring back the full list of students in the database.

By clicking the advanced search button the user can further refine the search value, the advanced search dialog is accessed by clicking the advanced search button. Advanced search allows the user to to search for students by id, name, phone, guardian or date of birth.

One the users desired student is found the user can click on the name in the main window and click a details button to view all the information for that student. In the details window the user is able to change the information for an entry and submit the updates to the system.

Add to Class:
This button brings up a window that allows the user to select a class. Once a class is selected the students currently registered will appear along with their class approval status. User can then select a student name

Figure 8.21: Student Schedule Window

and either click the add button to set the approval to one meaning the student is approved to take the class. A user can also remove the student which sets the approval to negative one meaning rejected. An approval of zero means the students registration is pending.

This function was created for the student interface links, however this set of functions were dropped so this function was left in at the request of the project's initial client Dr. Jeff McGough, so a future iteration of the project could see it. However in the projects current state this window serves no real overall purpose and should not be used by any users of the software. Anyone wishing to do student registration should do so through the student registration function explained below.

Student Schedule:
This button allows the user to pull up a window where they can select a student and view there class schedule. The user can also print out their schedule by pressing the print button. The schedule is design to only display the times that the student is in a class.
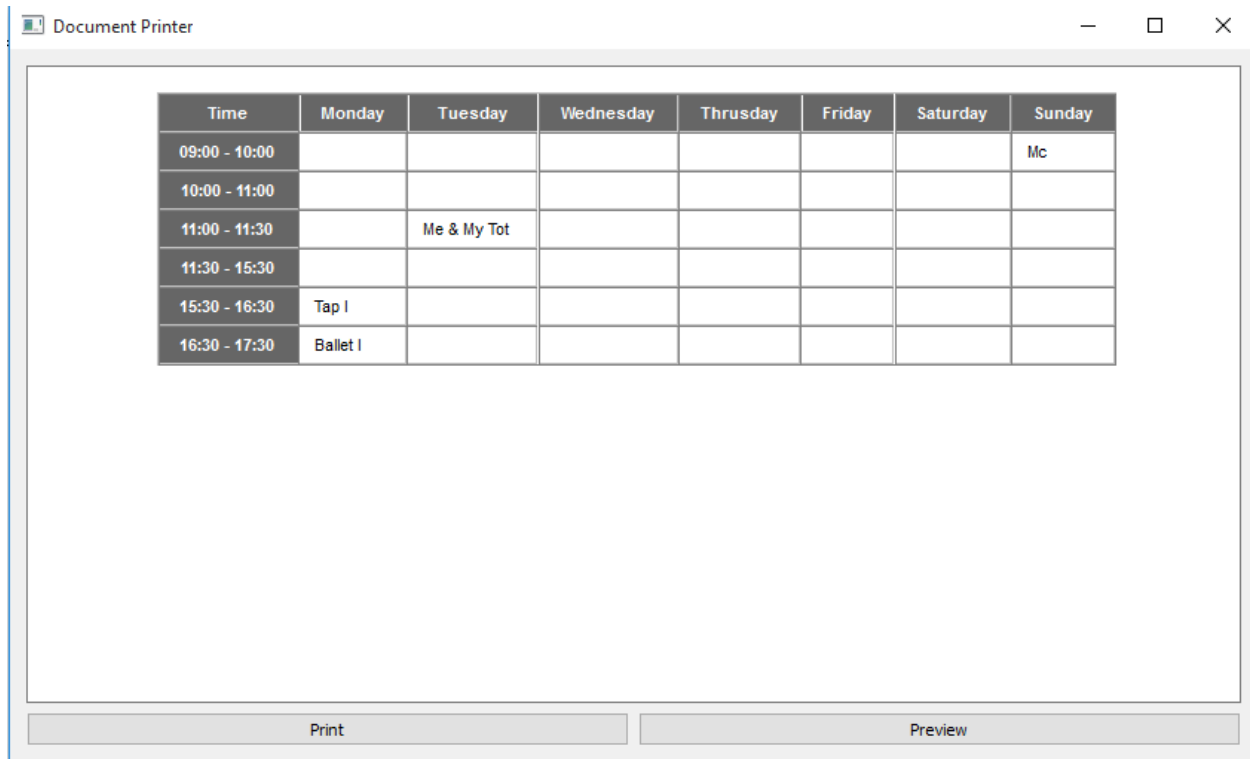
8.1.3.b   Class Options

This section contains the class related function that a teacher may need to manage their day to day activities.

View Classes:
This button allows the user to search classes like in the teacher search explained in the admin section. The general functionality is the same, the user is able to search by class name and refresh the view window after a search. The user can check the exact button if they wish to look only for what they entered. The Advanced search button allows the user to search for classes based on id, name location and time. Once the desired class is found the user can select a class and press the details button in order to view the full details for a course. Within this details page the user can modify the information and send updates to the system.

Class Schedule:

Figure 8.22: Teacher Schedule

This button allows the user to pull up a window where they can select a teacher and view there teaching schedule. The user can also print out their schedule by pressing the print button. The schedule is design to only display the times that the teacher is teaching a class.

Class Role Sheet:
This button creates a window which contains the classes the logged in teacher is currently teaching. The user can click on a class to view the students in the class. If the user wants a formatted role sheet they can click the print button which allows the user to print out a role sheet for the class. The role sheet contains the students name, and emergency contact information, along with a check box for each week in a three month period. This format is accordance with the one a week classes offer at the Academy of Dance Arts.

8.1.3.c   Personal Options

This section contains the functions connected to modify information directly connected to the user. These function include change password, change user name, modify personal information, and enter hours.
Change Password
When using this function the user is prompted with a dialog box, form this dialog the user will enter their user name, their desired new password, and confirm the new password. The user will be prompted to reenter their password if their user name or password are incorrect.

Change Username:
When using this function the user is prompted with a dialog box, form this dialog the user will enter their user name, their password, and the desired new user name. The user will be prompted to reenter their user name if their user name or password are incorrect.

Modify Personal Information:

Figure 8.23: Class Role Sheet

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e80a0c1c71763ff6b5a81f8cc9bb3d50	19435166	SIG
XZ compressed source tarball	Source release		8d526b7128affed5fbe72ceac8d2fc63	14307620	SIG
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later	8491d013826252228ffcdeda0d9348d6	24829047	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	349c61e374f6aeb44ca85481ee14d2f5	23170139	SIG
Windows debug information files	Windows		d6ffcb8cdabd93ed7f2feff661816511	37743788	SIG
Windows debug information files for 64-bit binaries	Windows		a0eea5b3742954c1ed02bddf30d07101	25038530	SIG
Windows help file	Windows		5fa4e75dd4edc25e33e56f3c7486cd15	7461732	SIG
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64, not Itanium processors	963f67116935447fad73e09cc561c713	26054656	SIG
Windows x86 MSI installer	Windows		e96268f7042d2a3d14f7e23b2535738b	24932352	SIG

Figure 8.24: Python zip files

If the user ever needs to change their personal information they can click this button to bring up a form containing their personal information. In this form the user can change their name, date of birth, phone numbers, address, and other information within the system. Like the other update teacher form in the admin section of the DanceSoft project, if the user leaves any of the required fields open they will be prompted to fill them in. Also if the user changes their address they will be given a dialog box asking if they would like to update their existing address or create a new one within the database. This is done because if two people going to, or working at the Academy live at the same address the teacher can create a new entry so that the other person address does not get effected. However if the teacher is the only one living at the address the system does not want to create dead data within the database, so the user is given an with an ability to change the entry.

As stated the user should use caution when ever changing address because if a user updates an address that someone else lives at then that person will have incorrect data, also if a teacher is the only person in the database with an address and a new address is added instead of updating then dead data will be created, and in this project iteration a user would need to access the database directly to remove a dead address as address removal is linked to teacher and student removal.

Enter Hours:
The user is able to look through the pay rates linked to their account by system admin and log hours for the various pay rates.

8.2 Installation Guide

The following is the steps to install the DanceSoft software on a users machine:

Step 1: The first thing a user needs to do is make sure that a valid version of Python 3 can run on their system, if you already have python 3 installed on your machine please skip to step 2. The most recent version of Python 3 can be found on `https://www.python.org/downloads/` the user can then click on the download link and download a python zip file that is compatible with the operating system being used by the user. Follow the instruction on the install, one the install is complete the python files should be located in your local drive unless the user specified a different directory during install. A user has several ways of confirming that python installed correctly on their system. If the user is with the command prompt they can run the Python 3 command to confirm successful installation. If the user would like to confirm using non-command line, the user can go to the python 3 file on their drive and run the Python.exe file. If a command window appears then the user has successfully installed python and can proceed to step 2.

Step 2: The next step is to install the PyQt libraries and the designer so the user can run the scripts containing PyQt code. There are several versions of PyQt, the version the DanceSoft team used was PyQt4. PyQt4 can be downloaded from `https://www.riverbankcomputing.com/software/pyqt/download`, one there the user can select the zip file that goes with their operating system. The website also provides

stable windows installers if the user is running a windows system. If the installer is used the libraries will automatically be stored in the site-packages sub-directory of the python folder and the user should be able to start using PyQt libraries If the user is not running windows, they can download and use the brew facility to easily install the need files. If the user runs **brew install PyQt --with-python3** from the OS X command line the system should automatically install the needed files. Otherwise the user will need to download the snippets from `https://www.riverbankcomputing.com/software/pyqt/download` and install the PyQt libraries in the site-packages folder.

Step 3: The last major install a user will need to do before using the software is the MySQL database software necessary to use the SQL components of the DanceSoft project. MySQL can be installed by following the download instructions on `https://www.mysql.com/downloads/` and downloading the free version of the software. This installer will install all the tools needed to manipulate and use the database directly if necessary.

The user should now have all the needed tools installed to run the DanceSoft Software.

9

Class Index

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

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 the Academy of Dance Arts 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, or 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 2016 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 DanceSoft senior design documentation 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 a 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 for this project.

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 are covered under the PyQt GPL license. In accordance with the South Dakota Board of Regents and the GPL license the project will be provided to the client and the PyQt code must be made open source if the client chooses to distribute the DanceSoft project beyond the client's personal use.

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.
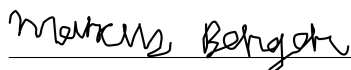
13 ATTACHMENTS

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

Marcus Berger (signature)   4/5/2016

Marcus Berger   Date

Dicheng Wu   4/5/16

Dicheng Wu   Date

Jeff McGough   4/5/16

Jeff McGough   Date

A

Product Description

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

NOTE: *This is part of the contract.*

B

Sprint Reports

---

1 Sprint Report #1

Sprint Report #1

1.1 Team Members:

Dicheng Wu
Marcus Berger
Sponsor:
Jeff McGough

1.2 Customer description

1.2.a 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.

1.2.b 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

1.2.c 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.

1.3 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.

1.4 Project Environment:

1.4.a 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

1.4.b 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.

1.5 Project deliverables of Sprint 1:

1. The research into program languages, database and GUI frameworks and architectures for the Dance-Soft 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.

1.6 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.

1.7 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

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

Figure B.1: DanceSoft Trello Board
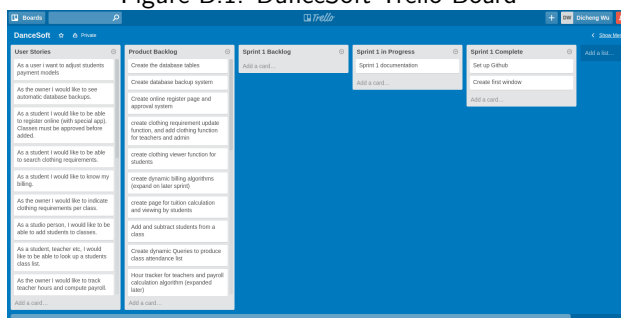
1.7.a Sprint 1 Backlog:

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

1.8 Research

1.8.a Database Research

1.8.b 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 fail overs. 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.

1.8.c 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.

1.8.d 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.

1.9 Final Framework Decision

Language: Python
Gui: PyQt
Database: Mysql

1.10 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

2.1 Team Members:

Marcus Berger
Dicheng Wu
Sponsor:
Jeff McGough

2.2　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.

2.3　Project deliverables of Sprint 2:

1. The database creation script

2. Gui path work, meaning figuring out the path through the Gui architecture

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)

2.3.a　Sprint 2 Backlog:

1. Creation of database tables

2. Draw GUI path

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

2.4　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.

2.5　GUI Work

2.5.a　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.

Figure B.2: Tables currently in database

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data
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	

Figure B.3: Current iteration of the log in page

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.

2.5.b 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 functionality 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 log in in page that takes a user name 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.

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

Figure B.4: Current iteration of the Admin Landing page
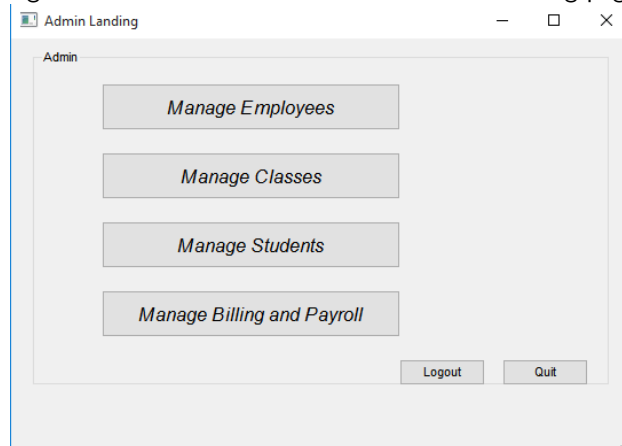
2. Modify data page

3. Add information and new information pages

2.6 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.

2.7 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

2.8 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.

2.9 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 #3

3.1 Team Members:

Marcus Berger
Dicheng Wu
Sponsor:
Jeff McGough

3.2 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.

3.3 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 B.5: Search Pages

3.3.a   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

3.4   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.

3.5   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

3.6   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.

3.7 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.

3.8 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.

3.9 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.

3.10 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

3.11 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.

3.12 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 #4

Sprint Report #4

4.1 Team Members:

Marcus Berger
Dicheng Wu
Sponsor:
Jeff McGough

4.2 Prototype Progress

During sprint 3.5 the team organized the functionality and tied them together to create the working prototype. Other updates were made to fix known bugs and add a class search. While the team worked on putting the project together smaller bugs or needed pages that were implied by user stories were tackled as they came up. This resulted in to much time being used, so the student interface was not tackled. The student interface has since been dropped from the clients project requirements due to time.

Sprint 4 was dedicated to payroll and billing. First the client rewrote the user stories to provide more clarity to the team. After this the user stories were tackled by the group to produce the first draft of the payroll and billing interface. This included logging hours, payments, fees, rates, credits, and discounts. After this was done the team demoed the project for the client and got notes on the project as a whole. These notes will be tackled as part of the backlog in the next sprint.

4.3 Project deliverables of Sprint 3.5 and 4:

Sprint 3.5 had no defined deliverables, and was mostly clean up.

Sprint 4 deliverable was the first draft of the payroll and billing interfaces and functions, layed out in the user stories below.

4.3.a Sprint 3.5 Backlog

1. Fix Assign Teacher to Class Dialog Bug

2. Add Class Search

3. Complete Role Sheet

4. Tie Functions Together

5. Fix Smaller Bugs

4.3.b Sprint 4 Backlog:

1. View Teaching History

2. Look at The Current Amount Someone Owes

3. Generate an Invoice For The Amount Due

4. Look at Billing History

5. Apply Credits to a Students Account

6. Enter the Tuition and Fees Rate

7. Give Early Registration Discounts

8. Enter Staff Pay Rates

9. Enter a Full Payment for One Student

10. Enter a Full Payment for Several Students

11. Enter Payments From Multiple Sources For One or More Students

12. Compute Teacher Wages

13. Enter Staff Hours

14. Give Prorated Refunds

4.4 Sprint 3.5

During sprint 3.5 Class Search was added this page follows the same format as other search pages created in previous sprints. Also completed fixes to the assign teacher to class bug, role sheet completeness, and a few other minors bugs. At the end of the sprint the team tied what we had together so the product could run as a unit starting from the log in screen.

4.5 Prorated Refunds

The prorated refunds code will keep track of how much a class costed a student. If the student drops a class it will refund the remaining worth of the class to the students school credit field. This file is still in progress.

4.6 Enter Staff Hours

This page needs to undergo changes after meeting with client. The new version will allow a teacher to log their in class hours, office hours, trip hours, etc. These hours will then be paid out at their different rates respectfully. The hours will be logged as single numbers not by days or weeks.

4.7 Enter Teacher Wages

This was handled in the database in previous sprints assuming one pay rate. However in talks with the client the academy pays different pay rates based on what part of academy work they are doing. So the page will need to be updated once the list of possible pay rates have been provided by the client.

4.8 Enter Tuition Rates and Fees

These pages allow the user to look at the current tuition rates and fees. The user can then update existing rates, add new rates, or remove rates that are no longer needed. Tuition rates are logged in the database as flat minute rates. The user can enter new rates in the form of minutes or hours.
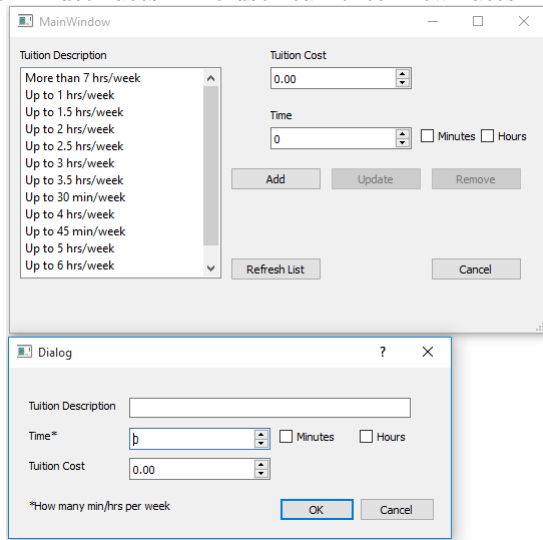
4.9 Teacher History

This page includes several components. Firstly, user can use this page to find a specific teacher by entering a full or partial name of teachers. Secondly, users can click the history button to open up another window which has a list of classes the teacher has taught.
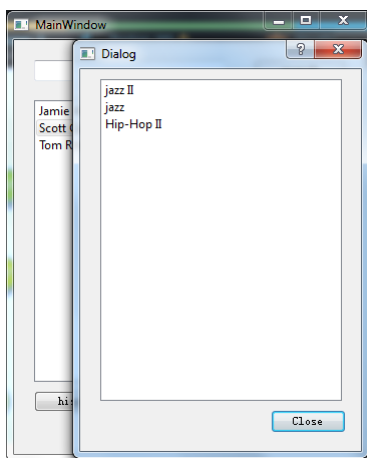

4.10 Set Semester

This page helps users to set the current semester from pool of semester and add a new semester to the system.
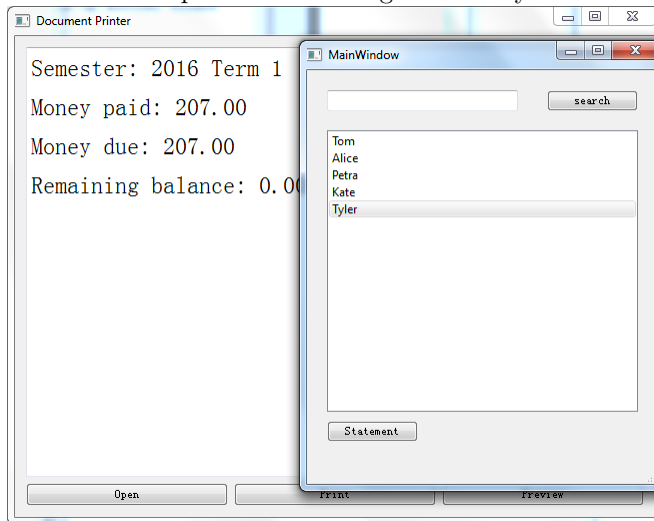
4.11   partial payment

This page includes several components. Firstly, user can use this page to find a specific student by entering a full or partial name of students. Secondly, users can click the pay button to open up another window which asks user to input amount of money paid, payment's method and semester paid. The user also can choose single or multiple students at one time.
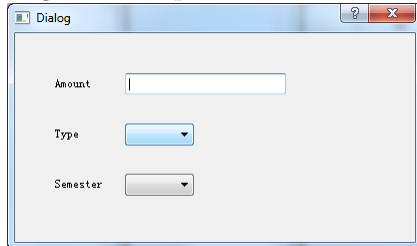
4.12   Student's owe

This page includes several components. Firstly, user can use this page to find a specific student by entering a full or partial name of students. Secondly, users can click the statement button to open up another window which shows the amount of students paid, the amount of students due and the balance of student's account. Users also can print the invoice generated by the window.

4.13   enter payment

This page includes several components. Firstly, user can use this page to find a specific student by entering a full or partial name of students. Secondly, users can click the clear button to clear the due of selected students. Users can select single or multiple students at one time.
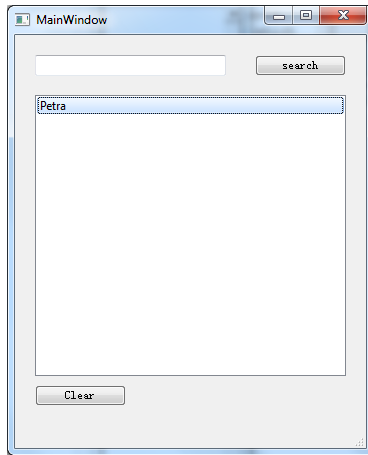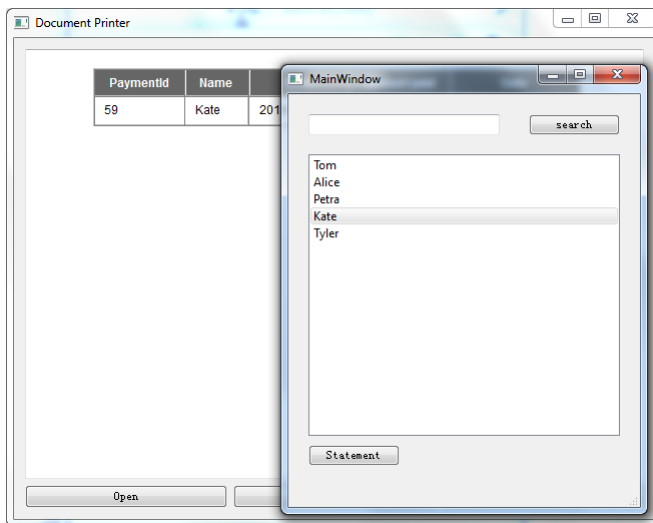
4.14   bill history

This page includes several components. Firstly, user can use this page to find a specific student by entering a full or partial name of students. Secondly, users can click the statement button in order to get the list of payments. In addition, user can print the bill history.

4.15   Sprint 3.5 and 4 Issues

Some issues that were encountered during this sprint were:

1. During winter break the team wanted to tackle the student interface app. Building other required pages in the main local GUI made this a problem. Therefore while some implied GUI pages where fixed and created, the team was unable to get to the student interface. As a result the student interface was removed from the project requirements.

2. Spring semester ramp up - Coming back from a month break meant that the team needed to ramp back up and get used to the new classes and new commitments. This cause the team to move slower then it should have in this sprint.

3. Team communication - The meeting this sprint were less effective as teams members would have other commitments, or in some cases lack the drive to do the work. This lessen as the sprint continued, but sometimes still came up.

4. Requirement Confusion - At times during this sprint the requirement were confusing. This resulted in a rewrite of the requires for payroll and billing to provide more clairity and adjust for dropping the student interface. The rework helped the team greatly, however the user stories still sometimes left out core part of how the academy does its work compared to other places. This resulted in spending more then the usually time speaking with the client, and group friction.

4.16   Client Interactions

Client interactions came in the form of requires reviews and a demo of the prototype at the end of the sprint. Topics included:

1. Progress reports on where we are in the project

2. Asking questions to refine functionality, understand academy needs, and update requirements.

3. Discuss the future of the project and idea on how to execute them.

4. Prototype the current version of the prototype.

4.17   Group Meeting

The group has a standard meeting time of TTH 10:00 - 12:00 and once during the weekend and at home as needed.

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.

4.18   Work Distribution

Marcus:

1. 3.5 Fix Assign Teacher Bug

2. 3.5 Update GUI Functions, Minor Additions and Fixes

3. 3.5 Tie Various Functions Into a Single Prototype

4. Prorated Refund (Still in Progress)

5. Enter Teacher Hours (Still in Progress)

6. Teacher Pay Rates (Needs Updated With New Information)

7. Early Registration Discounts

8. Gives Credits

9. Enter and Update Tuition Rates

10. Enter and Update Fee Rates and Type (Needs Updated With New Information)

11. documentation

12. Trello management

Dicheng:

1. 3.5 Tie function together

2. 3.5 Class Search Pages

3. 3.5 Updated Role sheet

4. 3.5 Research Linux Boxes

5. 3.5 Tie Functions Together Into a Single Prototype

6. Enter Payments From Multiple Sources For One or More Students

7. Look at The Billing History

8. Enter a Full Payment For One or More Students

9. Enter Payments From Multiple Sources For One or More Students

10. Generate an invoice for the amount due (Needs Updated)

11. Look at a The Current Amount a Student/Family Owes

12. View Teaching History

13. Auto and Manual discount giving

14. Set semester

Together:

1. Updated database construction

2. GUI page breakdown based on user stories and product backlog

3. Coded some functionality

5 Sprint Report #5

Sprint Report #5

5.1 Team Members:

Marcus Berger
Dicheng Wu
Sponsor:
Jeff McGough

5.2 Prototype Progress

Sprint five was dedicated to finishing the last of the functions and clean up of the project. First the client modified some requirements for the team. After this the user stories remaining from sprint four were tackled by the group to produce the remaining payroll and billing interface. This included logging hours, and payments. After this was done the team started on bug fixes and quality of life updates such as new buttons and removal functions. The team is currently working on finishing up the student registration, and tax calculation for wages. If progress continues at this rate the prototype should be completed functionally however without some work the prototype may not reach some of the teams desired standard.

5.3 Project deliverables of Sprint 5:

Sprint 5 deliverables were the remaining functionality for the project. If the sprint was successful the main project will be done or very close to done functionally at the end of the sprint.

5.3.a Sprint 5 Backlog

1. Admin and teacher update crossover

2. Ignoring case in address forms

3. Add rejected students to approved/ reject and provide current status

4. Multiple pay rates

5. Admin list function

6. Enter staff hours

7. Sprint 4 rollover (remaining payroll functions)

8. Add and remove class location

9. Removal Functions (Admin, Class, Student, Teacher)

10. Refunds and check boxes for fees and tuition

11. Quality of life updates to some functions

12. Bug Fixes

13. Student registration

14. Employee wages

15. Begin User Guide for documentation

5.4 Sprint 5

During sprint five the team tackled the remaining functionally, bugs, and quality updates in an attempt to finish the base project. Removal functions to clean out the database were added, and some client requested quality of life updates were added. Bugs and glitches were patch up in many functions. Another job of this sprint was to finish the rollover from sprint four since some payroll functions still needed work. Lastly the team tackled student registration since the student interface was dropped from the project requirements in sprint 4. The registration is still in progress at this time, as the team failed to complete this functionality in time for the end of the sprint.
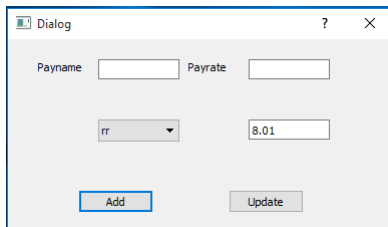
5.5 Prorated Refunds

The prorated refunds code will keep track of how much a class costed a student. If the student drops a class it will refund the remaining worth of the class to the students school credit field. This file has now been completed.

5.6 Enter Staff Hours

This page needed to undergo changes after meeting with client. The new version will allow a teacher to log their in class hours, office hours, trip hours, etc. These hours will then be paid out at their different rates respectfully. The hours will be logged as single numbers not by days or weeks. This function is now completed
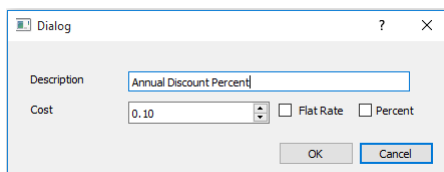
5.7 Enter Teacher Wages

This was handled in the database in previous sprints assuming one pay rate. However in talks with the client the academy pays different pay rates based on what part of academy work they are doing. So the page will need to be updated once the list of possible pay rates have been provided by the client. This pages and its updates are still in progress.

5.8 Enter Tuition Rates and Fees

Updated these pages to provide quality of life updates to the function requested by the client.

5.9 Bug Fixes

During the course of testing and the client meeting some bugs or inconsistencies were discovered within the project. The bugs found and fixed are:

1. Student search was not showing all the students. Status: Fixed

2. Search edit need to be turned off and advance search modifications. Status: Fixed

3. Search, role, and schedules need to have dynamic not static times. Status: Fixed

4. Allow for time changes on schedule. Status: Fixed

5. Some of the back buttons in the teacher interface did not work correctly. Status: Fixed

6. Forms seem to be bugged at times, however can not recreate bug. Status: In testing possibly fixed.

5.10 Updates Crossover

One of the updates that needed to be completed this sprint was the need for admin and teacher updates to crossover if something was changed. This means that if someone is an admin and they update there phone number through the "update my information form" then the phone number will be updated in both the teacher table and the admin table. This way the system will not have conflicting information in different places.

5.11 Approved/Rejected Student Updates

A update requested by the client was the ability to see rejected students in the approved/rejected pages just in case the academy changed its mind about a student. The page underwent slight modifications to the way information was displayed to make this change efficiently
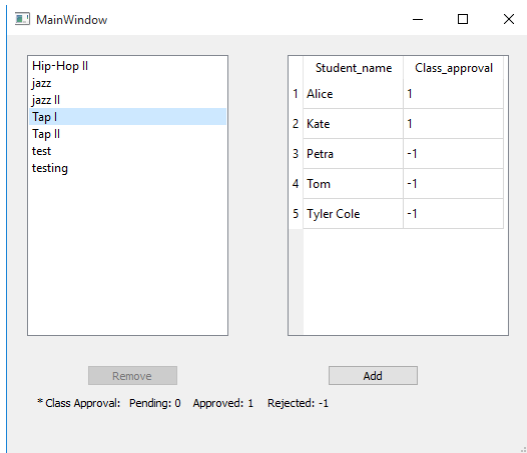
5.12　Admin List

In creating some of the updates it became clear that a specific admin list was needed to see who had admin access to the system. This was done through a simple list view of the names that can then be selected to display more detailed information.

5.13　Add/Remove Locations

Another requested feature was the ability to add and remove locations for classes in the database. This feature is necessary because the academy sometimes teaches classes in different places based on need. The team accomplished this using a simple dialog that connects to the location table in the database. When the user updates or adds a new class the location combo box provides a list of locations in the database and an add location option.
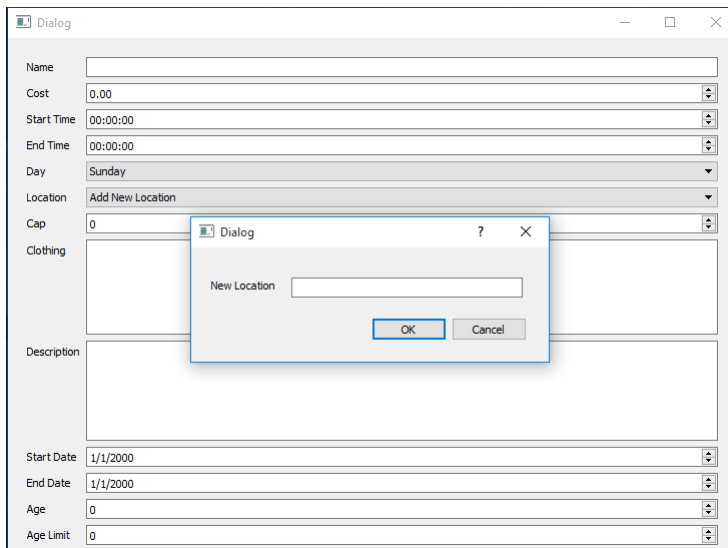
5.14　Removal Functions

These are simple functions that pull up a dialog box where the user can select a name and remove all the information related to that user from the system. The removal functions that exist in the system are: teacher, admin, student, and class. Teacher will remove that persons information from the system. Admin removes that persons admin rights and then ask if the whole user should be removed.
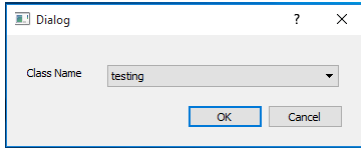
5.15 Sprint 5 Issues

Some issues that were encountered during this sprint were:

1. Spring semester ramp up - This was by far the biggest issue this sprint and the main cause of the failures this sprint. Other classes up to the point of blocking work on many projects. Staggered due dates meant that when the team wanted to work, there was always another projects that needed to be completed in a short amount of time. By the end much of our time had been taken and the sprint began to fall behind.

2. Requirement redefining - Due to drop the student interface the functionally needs to be added to the GUI which created some extra work for the team.

3. Failure to complete backlog - The team did not finish all of the assign backlog this sprint.

5.16 Client Interactions

Client interactions came in the form of required reviews and a demo of the prototype at the end of the sprint. Topics included:

1. Progress reports on where we are in the project

2. Asking questions to refine functionality, understand academy needs, and update requirements.

3. Discuss the future of the project and idea on how to execute them.

4. Prototype the current version of the prototype.

5.17 Group Meeting

The group has a standard meeting time of TTH 10:00 - 12:00 and once during the weekend and at home as needed.

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.

5.18 Work Distribution

Marcus:

1. Bug fixes

2. Removal functions

3. Admin and teacher update crossover

4. Ignore Case on address forms

5. Add rejected students to appoved/ reject and provide current status

6. Add/remove class location

7. Prorated refunds

8. Quality of life updates

9. Sprint report

10. Start User Guide

11. Trello management

Dicheng:

1. Bug fixes

2. Pay rates

3. Admin list

4. Staff hours

5. Student registration (in progress)

6. Wage calculations (in progress)

Together:

1. Updated database construction

2. Update prototype structure

3. Coded some functionality

C

Industrial Experience and Resumes

1 Resumes

Marcus J. Berger

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

Employment

Intern, SDSM&T University Relations**,** Rapid City, SD 　　　　　　　9/2013 to 9/2015
- 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
- Helped develop a project that won a Black Hills American Advertising Award

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 customers 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 2.86
- **Courses including but not limited to:** Assembly Language, Computer Graphics, Computer Networking, Database, and Data Mining
- **Current Major Courses:** Artificial Intelligence, Graphical User Interfaces, Operating Systems, and Senior Design
- **Projects include:** ANN, Database projects including PHP and SQL, Data Manipulation, Image Processers (C++ and ARM Assembly Language), Semi-Supervised Learning, Socket Programming, and a Solar System Model in OpenGL
- **Senior Design Project:** Developing an open source program, database and GUI to handle student information and other business information for the Academy of Dance Arts in Rapid City.

Qualifications

Professional Skills:
Experience with Agile development and Github
Knowledge of Waterfall development
Task Driven Development
Working in Teams
Oral and Written Communication

Software Skills:
Fluent in C++
Fluent in Python and PyQt
Experience with HTML, PHP and SQL
Lisp
Webpage Design (CMS, Google Analytics, etc.)
Some Java and Swing

References – Available Upon Request

Dicheng Wu

729 E Anamosa St. #101   (605) 389-6729

Rapid City, SD 57701   dicheng.wu@mines.sdsmt.edu

EDUCATION

South Dakota School of Mines and Technology - Rapid City, SD   **Overall GPA: 3.36**

- Computer Science B.S.   Major   GPA: 3.67
- Expected Graduation Date: May 2016

Related Courses
- Data Structures
- Analysis of Algorithms
- Software Engineering
- Database Management
- Assembly Language
- Digital Image Processing
- Programming Language
- Natural Computing
- Senior Design I
- Introduction to Robotics
- Data Mining & Visualization
- Networking/Data Communication

TECHNICAL SKILLS

- **Proficient:** C/C++, Python, JAVA
- **Experienced:** Common Lisp, SQL, ARM Assembly
- **Libraries:** CUDA, PyQt, NumPy, SciPy, Matplotlib
- **Operating Systems:** Windows, Linux
- **Database Systems:** MySQL, MSSQL
- **Environment:** Visual Studios, NetBeans, Qt Creator, LaTeX, Github, Agile Development

PROJECTS

- **Evolving Neural Networks To Play Tic-Tac-Toe:** The program is written in C++. The main functionalities of this program are to train a neural network by using genetic algorithm and to play tic-tac-toe with humans. I also rewrite the neural network with applying back-propagation and CUDA afterwards.
- **Dance Soft:** This is a senior design project. We are making an open-source software which will run on local Dance Academy for their students and classes' management. The software is written in Python and PyQt and built upon MySQL database. The development process strictly conforms to "agile development methodology".

PROGRAMMING CONTEST

- Placed 47th (out of 290 teams) at 2014 ACM-ICPC North Central North America Regional.

EXPERIENCE

- **South Dakota School of Mines and Technology Dining Service** - Jan 2015 – May 2015
  Cleaning tables and washing dishes.

2 ABET:Industrial Experience Reports

2.1 Marcus Berger

2.2 Dicheng Wu

D

Acknowledgment

The team would like to acknowledge the following people and groups from their input and assistance on various aspects of the DanceSoft project:

1. Brian Butterfield - For assistance in senior design class and project input

2. Dr. Mengyu Qiao - For assistance with database construction, and SQL references

3. Roger Schrader - For assistance in storing the database and computer access

4. Computers Unlimited Senior Design Group - For providing feedback and input on the project and the senior design class.

5. ARM Cluster Research Team - For providing feedback and input on the project and the senior design class.

6. Fellow Members of the Senior Design Class - For input, recommendations, and assistance during project development

E

Supporting Materials