1/24/2022

SPRINT TWO

CS2450-002, Team 2
Cody Strange-*Scribe and Information Manager*
Ethan Taylor-*GUI Developer*
Jaden Albrecht-*Team Manager*
Tyler Deschamps-*Chart and Milestone document builder*
Jordan Van Patten-*V&V and Tester*
Craig Sharp-*Stakeholder*

Table of contents

Introduction

Procedures

Document versioning

Summary: We currently have an archive folder in our google drive that we put any old documents into when we get a new one. Also in the title's of documents we put _v(x) for whatever version of that document is passed the initial version of the document.

Scheduling

Summary: We have the given deadlines from the customer for scheduling when our sprints need to be completed by. Jaden Albrecht is our team manager and schedules our meetings with the stakeholder. We have two meetings scheduled each week at 7pm on Monday and Friday.

Verification & Validation

Summary: We have processes for both verification and validation. Both are in early stages of implementation and are likely to change as we see what works and what doesn't as the sprints progress.

- Verification: We meet the Friday after the sprint begins and discuss our initial thoughts on the requirements of this sprint. We then have the weekend to mull over our ideas and figure out what we missed. Then we meet back on Monday and confirm what we agree needs to be done. We then schedule a meeting with the stakeholder so that we can get a final confirmation on what all needs to be done.
- Validation: We have a V&V folder that when we finish a document we submit it into that folder. Jordan then validates that the documents meet the proper requirements and are ready to be reviewed by the professor. Jordan then either sends them back to us to revise or to the review folder for the professor.

Timekeeping

Summary: We have the Gantt chart and the Burndown chart that shows the estimated time and the total time that is spent on the sprint, and tasks in the sprint. We also have a text message group that each day we send all the tasks we worked on and the amount of time we have spent on them.

1/24/2022

Requirements Gathering

Requirements

Summary: Requirements gathered from meeting with customer, creating MoSCoW charts, research. Requirements have been split between functional and non-functional. There are also the must-have requirements and the change orders that have been requested.

- Requirement Specifications:
  - Database Merging: recognize when there are incomplete fields and making a flag pop up that says "this is an incomplete employee"
  - Add employee: Page(B) of the GUI, Admin access, button on page(A), input all required fields and add employee to database
  - Edit employee: Page(C) of the GUI, General access, button on page(A), list of employee information to edit(First name, Last name, Address, Office phone, Personal phone, Bank info, Office email, Personal email)
  - Edit employee: Page(C) of the GUI, Admin access, button on page(A), list of employee information to edit(SS#, D.O.B, Pay type, Title Dept., Permission level)
  - Search employee by last name or ID: Page(A) of the GUI, General access, button on page(A), if user inputs numbers will search based on ID, if user inputs letters will search based off of last name, pulls up list of names and numbers of every that matches the information input
  - View employee: Page(A) of GUI, Admin access, list of employees names and IDs, when user clicks on name/ID pulls up all information related to that employee, option to view/hide deactivated employees
  - Deactivate employee: Page(D) of GUI, Admin access, button on page(A), input ID of employee that the user wants to deactivate, confirmation message pops up along with employee information to make sure the user wants to deactivate this specific employee
  - Win 10: The program will be able to run on Windows 10
  - Reports: Page(E) of GUI, Admin access, button on page(A), options to produce various reports(pay, reimbursables, employees)
  - Export Reports: Page(E) of GUI, Admin access, button on page(A), option to export any report as a csv
  - Secure records to only admin permissions: Certain information/records will require the user to be flagged as an Admin to see.
  - Intuitive GUI: Mouse over input boxes to see what information is required, help buttons on each page
  - User Manual: Page(F) of GUI, General access, help button on page(A), information on what each page can do
  - Readme.txt: A short description of what data the code contains
  - Simple just download installation that runs: Download software and then the user is good to go
  - Garbage proof entries: Checks each field of data to make sure that all information coming in isn't junk
  - Warn user of empty data fields: When user adds/edits an employee, issue a warning if any data fields are left empty or incomplete

    - Employee can view all personal fields: Page(A) of GUI, when employee logs in with his/her ID that ID's corresponding information is pulled up
    - Bug free: Software will go through extensive testing to minimize/eliminate as many bugs as possible
    - Requirements for employee information: First name, Last name, Address(use separated fields), Office phone, Personal phone, Emp ID(Specific length, only numbers), Pay type(commission, hourly, salary), D.O.B, SS#(Specific length, only numbers), Start date, End date, Bank Info(if Direct Deposit), Permission level, Title Dept., Office email, Personal email
- Functional
    - Add employee
    - Edit employee
    - Search employee
    - View employee
    - Deactivate employee
    - Reports
    - Export Reports
    - Secure records to only admin permissions
    - Warn user of empty data fields
    - Employee can view all personal fields
    - Requirements for employee information
- Non-functional
    - Win 10
    - Bug free
    - Garbage proof entries
    - Simple just download installation that runs
    - Readme.txt
    - User Manual
    - Intuitive GUI
    - Database merging
- Change orders
    - N/A

Prototype candidate

Jaden Albrecht

Summary: The program can read in a csv file of employee information, each employee is required to have an Id, first name, last name, Address, City, State, zip code, and pay classification. The program holds every employee's information in a list that can be accessed to look up an individual's information. The program also requires timecard, receipt, and pay log files in order to calculate each employee's pay. This is the prototype that we chose to use, because its code is more flexible out of the two prototypes we had.

1/24/2022

Current Features

- Calculate an employee's pay by their total salary, commission, or hourly wage
- Add employees to the database
- Find employees based on their ID number
- Reads in an csv file and out puts a list of employee objects
- Return specific employee information such as ID, name, address(city, state, zip code), classification(hourly pay, commissioned pay, salary pay)
- Alter an employee's classification

Needed Features

- Recognize when there are incomplete fields and make a flag pop up that says "this is an incomplete employee"
- Add an employee's:
  - First name
  - Last name
  - Office phone
  - Personal phone
  - Bank info
  - Office email
  - Personal email
  - SS#
  - D.O.B
  - Title Dept.
  - Permission level
- Search employees by last name
- Deactivate employees
- Export reports as a csv
- Secure records to only admin permissions
- GUI

Resources

Communication

Summary: We primarily use text messaging group to stay in contact with each other outside of planned meetings. We use it to ask for quick updates on tasks that may affect what we are currently doing. MS teams is what we use for our team meetings, we are currently planning to meet twice a week Monday and Friday at 7:00pm. There is also the project email and google drive that is being used to send documents to each other and to store files for the project.

- MS teams
- Text group chat
- Project email and Google drive

1/24/2022

Software

Summary: We are using Lucid Charts as the software to create the PERTT, Gantt, MoSCow, and Work Breakdown charts. Microsoft word is what we are using for documentation for our meeting logs, notes, and sprint documents. We are using Trello to help us organize a to-do list that is similar to Kanban. Python is the programming language that we will be working in, and we are still not sure whether we will be using Thonny or Visual Studio Code as the IDE. Tkinter will be used for the GUI. We will be using a group Gmail to send information to the group and Google Drive to save all of our documents so that the entire group and the shareholder will have access to them.

- Lucid Charts
- Microsoft word
- Trello
- Python
- Thonny/VS code
- Tkinter
- Debugging Software*
- Messages(Time keeping)
- Gmail
- Google Drive

Research

Sources

- Beginning Software Engineering(chapter 4)
- CS2450 Lecture #6(2/2/2022)
- Customer Meeting #2(2/1/2022)
- CS2450 Lecture #5(1/31/2022)

Agile Methodologies

Scrum

Overview

Summary: As the project sprints are designed around the Scrum methodology, we will be using Scrum in our project management. We will be completing our project in incremental iterations so that at the end of each sprint we will have a fully tested and approved piece of software. We still need to decide exactly what parts of Scrum will and won't work for our team.

- Scrum focuses on breaking large problems into smaller tasks and completing tasks with speed and efficiency.
- Scrum focuses on creating a small workable product soon and building new and more complex versions of that product every sprint.
- Scrum is flexible, allowing constant changes throughout development to meet customers everchanging requirements.

Crystal Clear

Overview

Summary: Along with Scrum will be incorporating part of the Crystal Clear development methodology. Crystal Clear like scrum focuses on frequent releases that get additional features as time goes on. Crystal Clear is very heavy on team communication, it recognizes that in small teams it is valuable to have everyone pitch in on ideas and that rigid structures are not necessary.

- Much more informal methodology
- Expected to deliver production every 2-3 weeks, possibly shorter with non-released development iterations
- Take into consideration what the program will be used for when deciding how strict certain requirements such as testing need to be
- Emphasis on communication and in person collaboration whenever possible

1/24/2022

Kanban

Overview

Summary: The last methodology that we will be using is Kanban, and we will be using Trello to implement this. We use the Kanban board to visualize what needs to be done, what is currently being worked on, and what has been completed.

- Kanban enables us to visualize the work that we are doing today in the context of other tasks
- Kanban board

Research

Sources

- Beginning Software Engineering(chapter 14)

Programming

Coding Standards

PEP-8

Summary: We will be following the PEP style guide for python code, as well as using either Pytest or Black to ensure that coding standards are followed.

Brainstorming

Team Meetings

Summary: We have our team meetings on Friday and Monday at 7pm. This allows us to come together on Friday to begin brainstorming and then we have the weekend to think over the ideas that were presented and meet back on Monday to finish the brainstorming.

High-Level Design

Plain English Code

Summary: The Plain English Code gives a description of our solution to the software we plan on creating in plain English that should be easy to read and understand. It goes over each of the screens that we plan on creating for the GUI and what needs to be done on each of them. It uses the Requirement Specifications as a guideline for the functionality planned for each window. The Plain English Code does not state specifics on how each GUI window will be created/implemented only what needs to be done on each one.

Login Screen:

- Import Tkinter
- Set up main window (can be inherited by other subsequent pages/windows)
- Set title and labels for user inputs (username and password)
- Set up grid manager to control widget size and placement
- Get username and password from user inputs
- Check for provided username in the database
- If username not in database, report username is not valid employee
- If username or password incorrect, display warning
- Validate username and password, get user permission level, and display proper window based on permission level

View Employee Screen (Admin):

- Set up view screen window (can be inherited by non-admin permission screen, as well as Add Employee and Edit Employee since they use the same labels)
- Set title and labels with accompanying entries for employee attributes (first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status)
- Set up buttons in a sidebar for edit employee(opens edit employee screen), add employee(opens add employee screen), deactivate employee(opens deactivate employee screen), search database(opens search screen), print payroll(outputs payroll info as a csv file), and help(opens user manual)
- Set up grid manager to control widget size and placement
- Get employee object from database using provided username and password
- Use attributes of employee object to set the various fields so that by default the user is looking at their own info
- Set entry fields to read-only
- If employee is missing any required fields, display a warning

View Employee Screen (Non-admin):

- Inherit window and grid properties from admin version of screen
- Restrict employee attribute labels (first name, last name, address, city, state, zip, office phone, office email, title, dept, employment status)
- Restrict sidebar buttons to just edit employee, search database, and help
- Get employee object from database using provided username and password
- Use attributes of employee object to set the various fields so that by default the user is looking at their own info
- Set entry fields to read-only
- If employee username does not match name in employee object, disable edit button (so a non-admin employee can only edit themselves)
- If employee is missing any required fields, display a warning

Edit Employee Screen:

- Inherit window and grid properties from admin version of screen

1/24/2022

- Set screen title and labels with accompanying entries for all employee attributes (first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status)
- Set up a "save-changes" button that will return the user to the view screen and help button
- Get employee object from database
- Use attributes from employee object to fill out entry fields
- If employee is missing any required fields, display a warning
- Validate entry fields with specific requirements (Emp ID(specific length, only numbers), SS#(specific length, only numbers))
- Upon click of "save-changes" button, update database and return to view screen

Add Employee Screen:

- Inherit window and grid properties from admin version of screen
- Set screen title and labels with accompanying entries for all employee attributes (first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status)
- Set up button for add employee and help button
- Validate entry fields with specific requirements (Emp ID(specific length, only numbers), SS#(specific length, only numbers))
- Upon click of "add" button, build new employee object and add it to database, then return to view screen

Deactivate Employee Screen:

- Set up screen window
- Set screen title and label with accompanying entry field for Emp ID
- Set up "deactivate employee" button and help button
- Check if entered Emp ID is in database
- If Emp ID not in database, report Emp ID invalid
- If valid Emp ID, on click of "deactivate" button ask for confirmation
- On confirmation set employee status to deactivated and return to view screen

Search Employees Screen:

- Set up search screen window
- Set up combo-box** with list of search parameters (Must have last name and Emp ID, others optional)
- Set up entry field for search parameters and help button
- Set search entry to read-only until search parameter chosen
- Set up grid geometry to control widget size and placement
- Apply a filter to the entry field that will search the database for potential matches and provide results of this search in a drop-down list as the user types*
- If search returns no matches, report no match found
- Upon selection of searched employee, take employee info to view employee screen

Help Pop-Up Screen:

- On hover over important fields, display a small window with information about that field*

Employee Class:

- Employee object has following attributes: first name, last name, address, city, state, zip, office phone, personal phone, Emp ID, pay type/classification, D.O.B, SS#, start date, bank info(if Direct Deposit), permission level, title, dept., office email, personal email, employment status
- Methods to find employee object by Emp ID and last name, possibly more**
- Methods to change employee attributes

Global Functions:

- Function to read in employee csv into list
- Functions to read in timecard and receipt csv files
- Function to process and output payroll report

Security

Summary: This early on, we have not confirmed what security measures we plan on using. Though we are looking into some basic data encryption that Visual Studio Code offers.

Hardware

Summary: The software will be designed to run on a laptop or personal computer for windows 10. Support for mobile devices is not currently planned.

External Interface

Summary: We are currently using Tkinter for creating the GUI and as we get closer to the coding process we may decide to use GitHub to store our code.

Internal Interface

Summary: The GUI classes will use the properties of the Employee class to fill out the data fields on the GUI screen.

Reports/Outputs

Summary: Our current prototype from CS1410 has the ability to output pay reports to txt files, this will be expanded in our final product to report to output to csv files.

Database Design and Issues

Summary: Our database design is going to be in text, because we are going to be using csv files.

Training

Summary: The user manual will be in a help button on the GUI that will explain to the user how to use the software.

1/24/2022

Risk Management

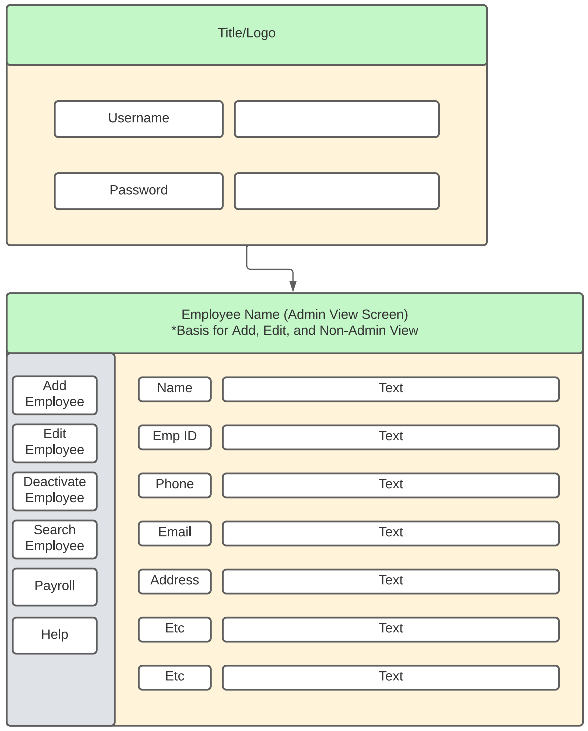
Summary: Risk management is accomplished through using the V&V processes and using debugging software and proper coding standards.

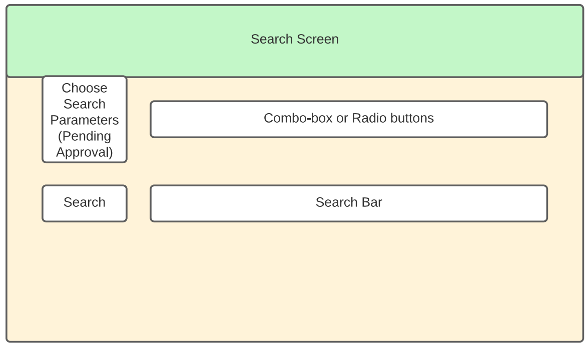
Quality Control

Summary: When someone finishes a section of code for the project another teammate will test and verify the code to guarantee that the code runs properly and follows the proper coding standards.

GUI Design

Summary: First version of the template for the GUI we will be creating. Shows the different screens that'll be implemented, and the information shown on each screen. Does not yet have the screen for reports or for deactivating employees




Search Screen
Choose Search Parameters (Pending Approval)
Combo-box or Radio buttons
Search
Search Bar

Architecture

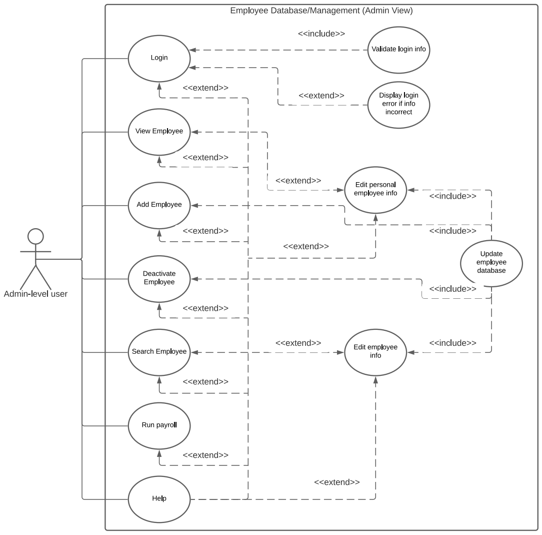
Summary: We will be using a monolithic structure as our architecture

- A single program does everything
  - It displays the user interfaces
  - Accesses data
  - Processes customer orders

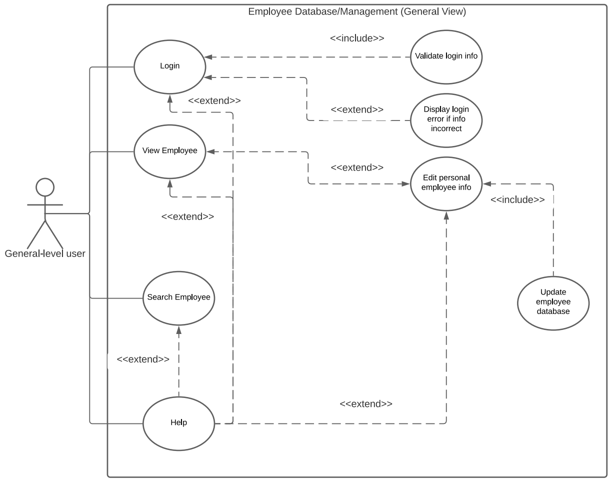
Use Case Scenarios

Summary: The use case scenarios are based on the privileges that the user has, whether they are an admin user or a general employee user. The admin user can add employees, edit employees(including themselves), deactivate employees, print out report, and search employees based on id or last name. The general user can edit employees(themselves), view employees(limited fields). Both have access to the help screen, and both will utilize the login screen.

- Admin Use-Case



- General Use-Case



Docstring Code Outline

Summary: The docstring code outline outlines the components of the software that will be programmed. We have an employee class that creates employee objects that are classified by their pay type(hourly, salary, commission). We have Tkinter based GUI classes that will control the various screens of the components.

```
#Import any needed modules
#Initiate constants
Employee Class:
“Creates Employee object”
        #Establish classification object
        #Getters (to fill out view screen)
        “Return employee ID”
        “Return employee first name”
        “Return employee last name”
        “Return employee address”
        “Return employee city”
        “Return employee state”
        “Return employee zip”
        “Return employee classification”
        “Return employee salary”
        “Return employee commission rate”
        “Return employee hourly rate”
        “Return employee office phone”
        “Return employee personal phone”
        “Return employee office email”
        “Return employee personal email”
        “Return employee D.O.B”
        “Return employee SS#”
        “Return employee direct deposit info”
        “Return employee permission”
        “Return employee title”
        “Return employee dept”
        “Return employee employment status”
        “Return employee start date”
        “Return employee end date”

        #Setters (to use with edit screen)
        “Set employee ID”
        “Set employee first name”
        “Set employee last name”
        “Set employee address”
        “Set employee city”
        “Set employee state”
        “Set employee zip”
        “Make employee salaried”
        “Make employee commissioned”
```

“Make employee hourly”
“Set employee office phone”
“Set employee personal phone”
“Set employee office email”
“Set employee personal email”
“Set employee D.O.B”
“Set employee SS#”
“Set employee direct deposit info”
“Set employee permission”
“Set employee title”
“Set employee dept”
“Set employee employment status”
“Set employee start date”
“Set employee end date”

#Other methods
“Pays the employee”
“String representation of employee object”

Classification Class:
“An abstract class of a classification object for an employee”
"Abstract decorator method of computing an employee's payment"

Hourly Class:
"Class that creates an hourly classification"
"Adds timecards to the list of timecards"
"Computes the payment of an hourly employee"

Salaried Class:
"Class that creates a salaried classification for an employee"
"Computes the payment of a salaried employee"

Commissioned Class:
"Constructs a commission classification for an employee"
"Adds receipts to the list of receipts"
"Computes the payment for a commissioned employee"

#Global Functions
"A function that processes employees database file into a useable list"
"Adds receipts to every commissioned employee"
"Adds timecards to every hourly employee"
"Finds an employee by their ID number"
"Finds an employee by their last name"
"Pays each employee"

#GUI Classes
Login Class:
“Controls the login screen of the GUI”

Admin_View Class:
“Controls the admin view screen of the GUI”

#Show all fields and buttons
#On click of deactivate button, show extra screen requesting confirmation

General_View Class:
“Controls the general view screen of the GUI”
#Only show first name, last name, address, city, state, zip, office phone, office email, title, dept, employment status
#Only show edit button if username equals employee, else show search and help buttons only

Edit_Screen Class:
“Controls the edit employee screen of the GUI”

Add_Screen Class:
“Controls the add employee screen of the GUI”

Search_Screen Class:
“Controls the search screen of the GUI”
#If user enters numbers search by EmpID, else if user enters letters search by last name

Testing

Summary: For testing the most likely software we plan on using will be Pytest with Assertion testing. May be using Black as well for making sure the program follows the proper coding standards.

Research

Sources

- Python.org
- Beginning Software Engineering(chapter 5)
- Modern Tkinter
- Course material

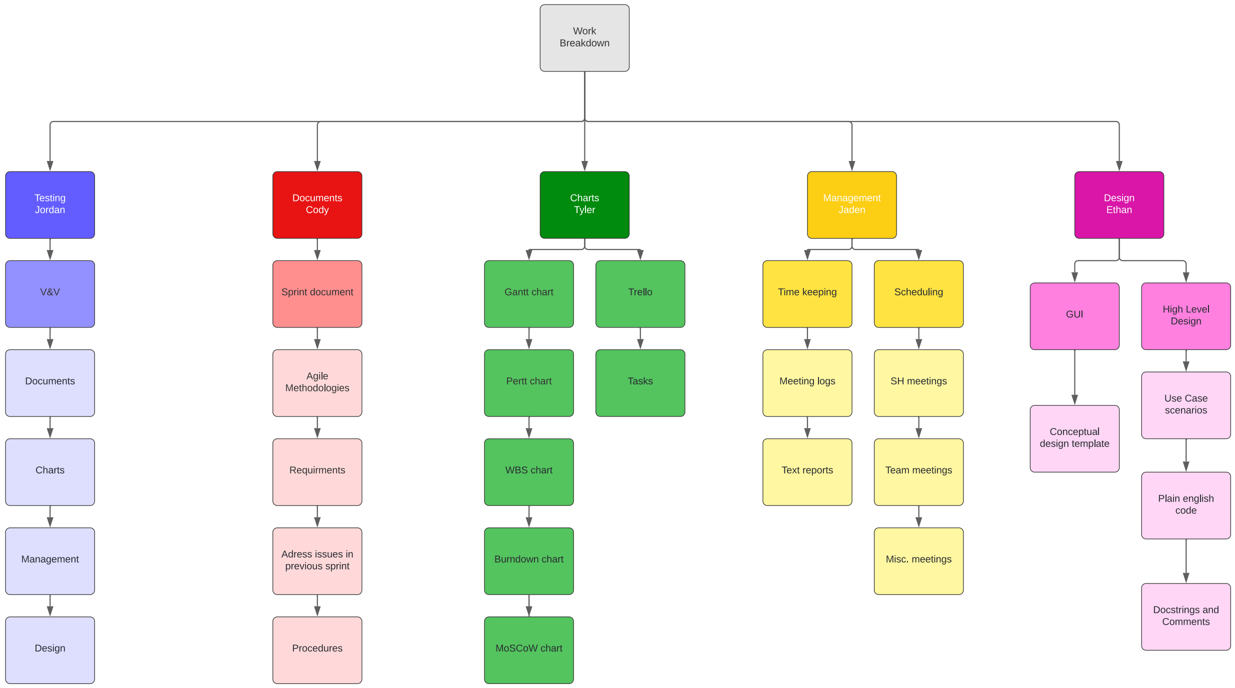
1/24/2022

Charts/Templates

Work Breakdown Structure

Template

Summary: This is a Work Breakdown Structure for our five members. It details each major tasks that need to be completed and who is in charge of managing said tasks, making sure that they are completed.

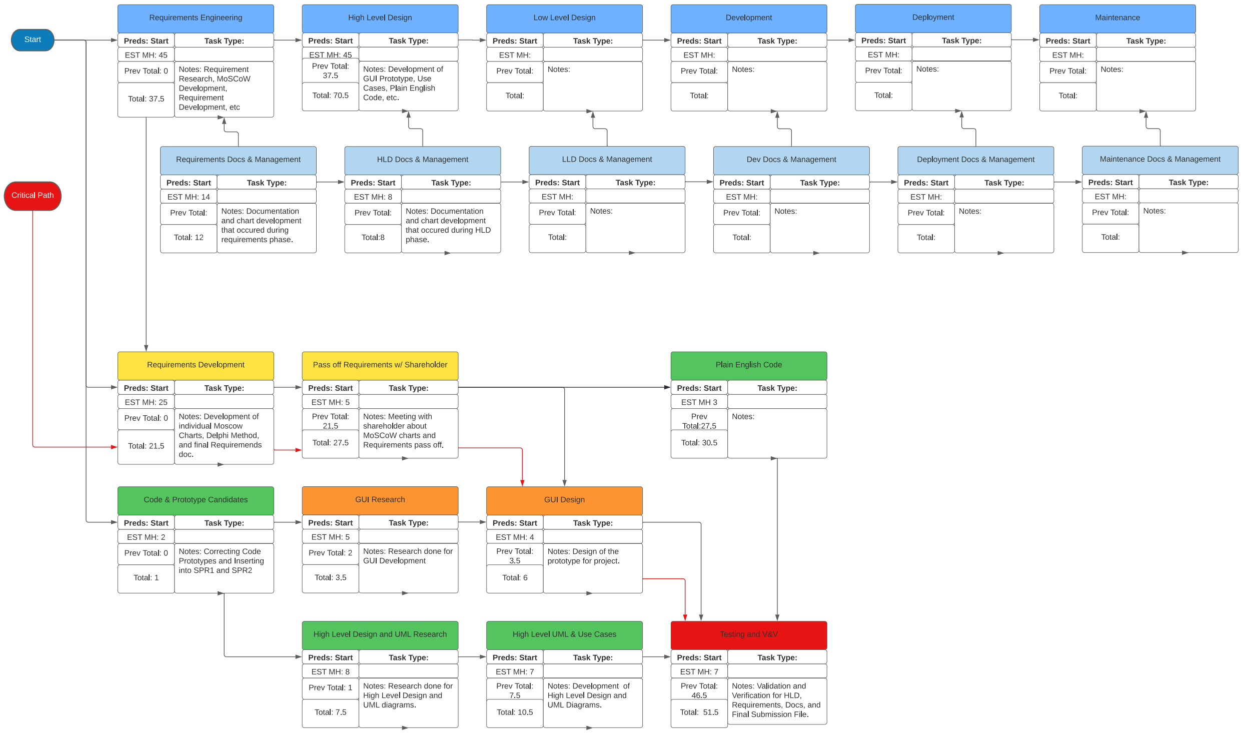


1/24/2022

PERT Chart

Template

Summary: This is a PERT Chart Template, for each of the stage. It has the estimated time we expect each task to take and come up with a total amount of time for the project. The critical path come out to 41hours



Requirements Engineering
Preds: Start | Task Type:
EST MH: 45
Prev Total: 0
Total: 37.5
Notes: Requirement Research, MoSCoW Development, Requirement Development, etc

High Level Design
Preds: Start | Task Type:
EST MH: 45
Prev Total: 37.5
Total: 70.5
Notes: Development of GUI Prototype, Use Cases, Plain English Code, etc.

Low Level Design
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Development
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Deployment
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Maintenance
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Requirements Docs & Management
Preds: Start | Task Type:
EST MH: 14
Prev Total:
Total: 12
Notes: Documentation and chart development that occured during requirements phase.

HLD Docs & Management
Preds: Start | Task Type:
EST MH: 8
Prev Total:
Total:8
Notes: Documentation and chart development that occured during HLD phase.

LLD Docs & Management
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Dev Docs & Management
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Deployment Docs & Management
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Maintenance Docs & Management
Preds: Start | Task Type:
EST MH:
Prev Total:
Total:
Notes:

Requirements Development
Preds: Start | Task Type:
EST MH: 25
Prev Total: 0
Total: 21.5
Notes: Development of individual Moscow Charts, Delphi Method, and final Requirements doc.

Pass off Requirements w/ Shareholder
Preds: Start | Task Type:
EST MH: 5
Prev Total: 21.5
Total: 27.5
Notes: Meeting with shareholder about MoSCoW charts and Requirements pass off.

Plain English Code
Preds: Start | Task Type:
EST MH 3
Prev Total:27.5
Total: 30.5
Notes:

Code & Prototype Candidates
Preds: Start | Task Type:
EST MH: 2
Prev Total: 0
Total: 1
Notes: Correcting Code Prototypes and Inserting into SPR1 and SPR2

GUI Research
Preds: Start | Task Type:
EST MH: 5
Prev Total: 2
Total: 3.5
Notes: Research done for GUI Development

GUI Design
Preds: Start | Task Type:
EST MH: 4
Prev Total: 3.5
Total: 6
Notes: Design of the prototype for project.

High Level Design and UML Research
Preds: Start | Task Type:
EST MH: 8
Prev Total: 1
Total: 7.5
Notes: Research done for High Level Design and UML diagrams.

High Level UML & Use Cases
Preds: Start | Task Type:
EST MH: 7
Prev Total: 7.5
Total: 10.5
Notes: Development of High Level Design and UML Diagrams.

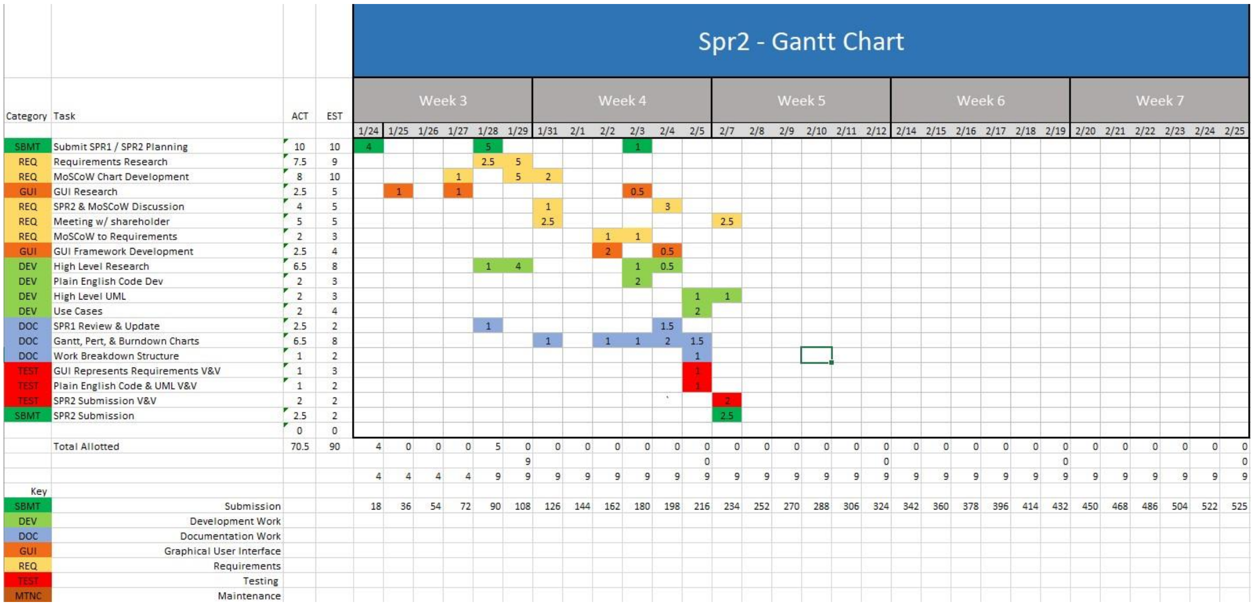
Testing and V&V
Preds: Start | Task Type:
EST MH: 7
Prev Total: 46.5
Total: 51.5
Notes: Validation and Verification for HLD, Requirements, Docs, and Final Submission File.

1/24/2022

Gantt Chart

Template

Summary: This is a Gantt chart template; it will list each of the tasks that we will be doing in each sprint, with the estimated time and actual time for each task.



Spr2 - Gantt Chart

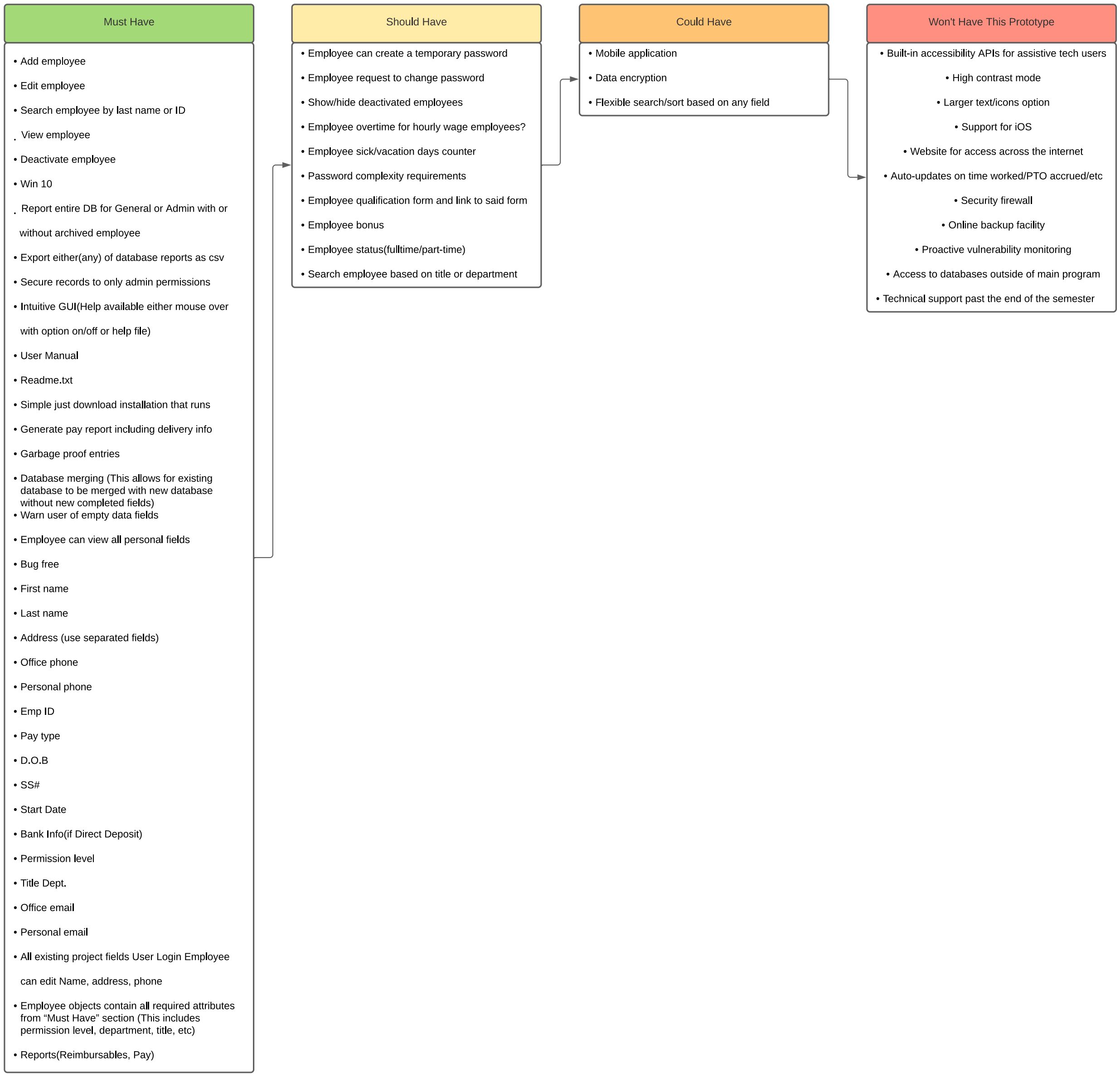
				Week 3						Week 4						Week 5						Week 6						Week 7					
Category	Task	ACT	EST	1/24	1/25	1/26	1/27	1/28	1/29	1/31	2/1	2/2	2/3	2/4	2/5	2/7	2/8	2/9	2/10	2/11	2/12	2/14	2/15	2/16	2/17	2/18	2/19	2/20	2/21	2/22	2/23	2/24	2/25
SBMT	Submit SPR1 / SPR2 Planning	10	10	4				5					1																				
REQ	Requirements Research	7.5	9					2.5	5																								
REQ	MoSCoW Chart Development	8	10				1		5	2																							
GUI	GUI Research	2.5	5		1		1						0.5																				
REQ	SPR2 & MoSCoW Discussion	4	5							1				3																			
REQ	Meeting w/ shareholder	5	5							2.5						2.5																	
REQ	MoSCoW to Requirements	2	3									1	1																				
GUI	GUI Framework Development	2.5	4									2		0.5																			
DEV	High Level Research	6.5	8					1	4				1	0.5																			
DEV	Plain English Code Dev	2	3										2																				
DEV	High Level UML	2	3												1	1																	
DEV	Use Cases	2	4												2																		
DOC	SPR1 Review & Update	2.5	2					1						1.5																			
DOC	Gantt, Pert, & Burndown Charts	6.5	8							1		1	1	2	1.5																		
DOC	Work Breakdown Structure	1	2												1																		
TEST	GUI Represents Requirements V&V	1	3												1																		
TEST	Plain English Code & UML V&V	1	2												1																		
TEST	SPR2 Submission V&V	2	2													2																	
SBMT	SPR2 Submission	2.5	2													2.5																	
		0	0																														
	Total Allotted	70.5	90	4	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									9						0						0						0						0
				4	4	4	4	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Key																																	
SBMT	Submission			18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360	378	396	414	432	450	468	486	504	522	525
DEV	Development Work																																
DOC	Documentation Work																																
GUI	Graphical User Interface																																
REQ	Requirements																																
TEST	Testing																																
MTNC	Maintenance																																

1/24/2022

MoSCoW Chart

Template

Summary: This is a MoSCoW chart template; it will have each team member listing what they believe the product Must have, Should have, Could have, and Won't have(this prototype).



Must Have

- Add employee
- Edit employee
- Search employee by last name or ID
- View employee
- Deactivate employee
- Win 10
- Report entire DB for General or Admin with or without archived employee
- Export either(any) of database reports as csv
- Secure records to only admin permissions
- Intuitive GUI(Help available either mouse over with option on/off or help file)
- User Manual
- Readme.txt
- Simple just download installation that runs
- Generate pay report including delivery info
- Garbage proof entries
- Database merging (This allows for existing database to be merged with new database without new completed fields)
- Warn user of empty data fields
- Employee can view all personal fields
- Bug free
- First name
- Last name
- Address (use separated fields)
- Office phone
- Personal phone
- Emp ID
- Pay type
- D.O.B
- SS#
- Start Date
- Bank Info(if Direct Deposit)
- Permission level
- Title Dept.
- Office email
- Personal email
- All existing project fields User Login Employee can edit Name, address, phone
- Employee objects contain all required attributes from "Must Have" section (This includes permission level, department, title, etc)
- Reports(Reimbursables, Pay)

Should Have

- Employee can create a temporary password
- Employee request to change password
- Show/hide deactivated employees
- Employee overtime for hourly wage employees?
- Employee sick/vacation days counter
- Password complexity requirements
- Employee qualification form and link to said form
- Employee bonus
- Employee status(fulltime/part-time)
- Search employee based on title or department

Could Have

- Mobile application
- Data encryption
- Flexible search/sort based on any field

Won't Have This Prototype

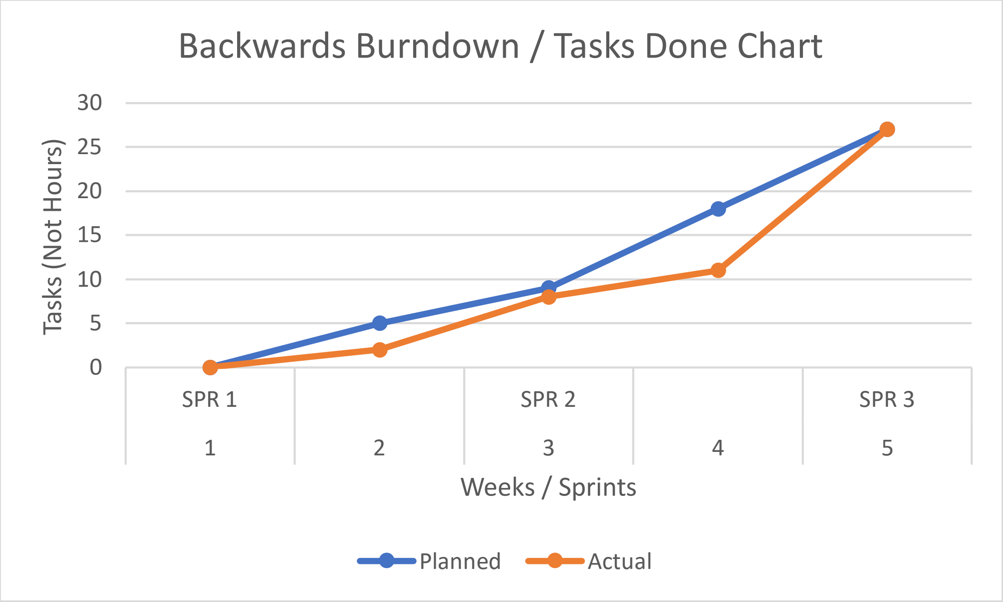
- Built-in accessibility APIs for assistive tech users
- High contrast mode
- Larger text/icons option
- Support for iOS
- Website for access across the internet
- Auto-updates on time worked/PTO accrued/etc
- Security firewall
- Online backup facility
- Proactive vulnerability monitoring
- Access to databases outside of main program
- Technical support past the end of the semester

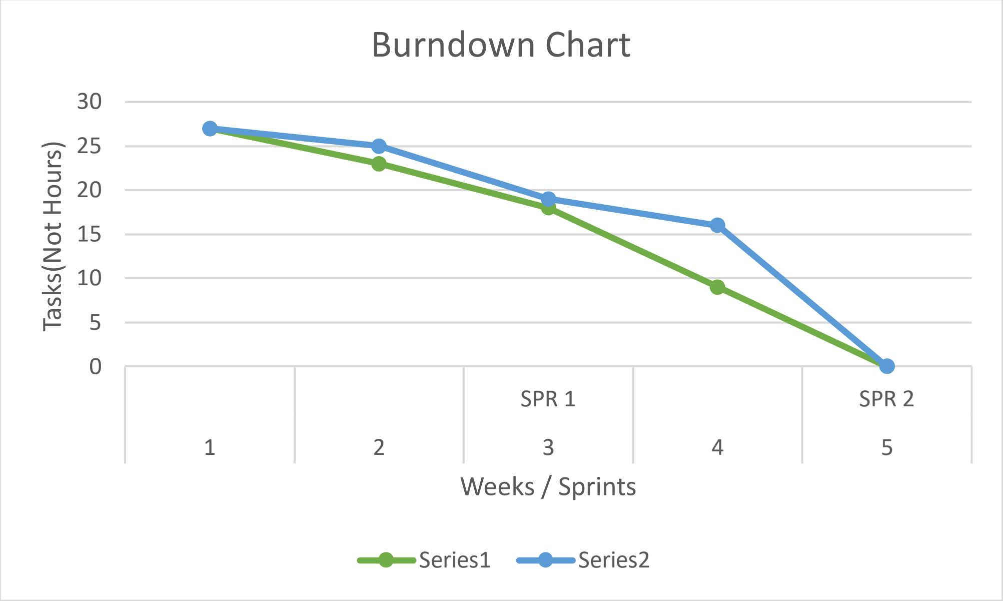
1/24/2022

Burndown Chart

Template

Summary: The burndown chart shows the amount of time expected to be spent on tasks in the sprint against the actual amount of time spent. There are two versions of the chart, the regular burndown chart, and the backwards burndown chart.





Research

Sources

- Beginning Software Engineering(chapter 3)
- CS2450 Lecture #6(2/2/2022)

Meeting Logs

Meeting Log#4

Team #: T2-002
Meeting log #: 4
Date: January 28, 2022
Time: 7:00pm – 8:05pm
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Current Sprint: SPR2

Sprint Progress

Ethan Taylor:

- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Jaden Albrecht:

- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Cody Strange:

- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Tyler Deschamps:

- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)
- Created Trello board for task management

Jordan Van Patten:

- Read all SPR2 documentation
- Finished personal MoSCoW chart (30 minutes)

Topics Discussed

- Next shareholder meeting
- MoSCoW charts (Delphi method)
- Testing methods
- Task assignment
- Preparations for high-level design

Obstacles Encountered

- No obstacles encountered

Finished Items

- All team members have thoroughly read and grasped the requirements for SPR2

- Each team member constructed individual MoSCoW charts based on user requirements discussed in chapter 4
- Assigned Jordan as tester/V&V/QA consultant
- Scheduled next team meeting for January 31, 2022

Unfinished Items

- Final requirements spec
- High-level design (plain English description)
- Create classes for GUI
- Use-case scenarios for high-level design
- Develop pert, Gantt, work breakdown structure, and MoSCoW charts
- SPR1 prototype candidate
- Establish connection between tasks and requirements
- V&V
- Schedule next shareholder meeting

Tasks Until Next Meeting

- Jaden is responsible for reviewing each individual MoSCoW chart and combining them into two separate charts
- Each chart will be distributed to one of two subgroups of the team and will be reviewed by each subgroup
- The final MoSCoW chart will be assembled after both subgroups have made their changes to each chart

Notes

- There was not a whole lot to work with at this point in the development cycle
- Once the final MoSCoW chart is assembled, each team member will have a clearer understanding of what is required of them throughout the remainder of the current sprint

Meeting Log#5

Team #: T2-002
Meeting log #: 5
Date: January 31, 2022
Time: 6:55pm – 8:10pm
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps, Jordan Van Patten
Current Sprint: SPR2

Sprint Progress

Jaden Albrecht:

- Distribution of MoSCoW charts for Delphi (2 hours:55 minutes)

Topics Discussed

- Final MoSCoW chart
- Next team meeting

1/24/2022

- Beginning high-level design
- Beginning requirements spec
- Begin development of pert, Gantt, and work breakdown structure (WBS) charts
- Methods for tracking tasks and requirements
- V&V

Obstacles Encountered

- Jaden is unable to edit or create new cards on team Trello board

Finished Items

- Distributed MoSCoW for Delphi
- Added Jordan to team Google drive, Trello board, and text chat
- Submitted list of disapprovals from shareholder from SPR1
- Scheduled next team meeting for February 4, 2022
- Scheduled next shareholder meeting for February 1, 2022

Unfinished Items

- Requirement spec
- Plain-English description of high-level design
- Classes for GUI
- Use-case scenarios for high-level design
- V&V
- Burndown chart
- Up-to-date pert/Gantt charts
- Schedule next shareholder meeting

Tasks Until Next Team Meeting

- Finalize MoSCoW chart
- Read SPR2 documentation
- Work on plain-English description for high-level design
- Develop classes for GUI
- Develop use-case scenarios for GUI
- SPR1 prototype candidate
- Create requirement spec
- Establish connection between requirements and tasks
- Update meeting logs to new template
- Begin developing burn-down chart
- Develop Gantt chart
- Logging for Gantt/pert charts
- V&V covering requirements and documentation
- Continue referring to chapters 4-6 for more help

Notes

- At this point, we have our final MoSCoW chart complete and ready to discuss with the shareholder
- From now until our next team meeting (not with the shareholder), Tyler will begin development of burn-down chart and begin tracking minutes on Gantt chart
- Cody will make final changes to MoSCoW chart according to preferences of the shareholder and will also begin the requirement spec

1/24/2022

- Ethan, Jaden, and Jordan will research tkinter and begin high-level design

Meeting Log#6

Team #: T2-002
Meeting log #: 6
Date: February 4, 2022
Time: 7:05pm – 8:00pm
Location: MS Teams (watch video here)
Attendees: Ethan Taylor, Jaden Albrecht, Cody Strange, Tyler Deschamps
Current Sprint: SPR2

Sprint Progress

Ethan Taylor:

- Review requirements for SPR2
- High-level design research (1 hour)
- Plain English description for high-level design (2 hours, 5 minutes)
- GUI template (30 minutes)
- Develop use-case scenarios based on requirements (1 hour, 45 minutes)

Jaden Albrecht:

- Review requirements for SPR2
- Review plain-English description of high-level design (30 minutes)
- Establish connection between requirements and tasks (1 hour)
- Changes to meeting logs and developed method for minute tracking (3 hours:30 minutes)
- Plan for next team meeting (5 minutes)

Cody Strange

- Review requirements for SPR2
- Create requirements spec (1 hour)
- Changes to final MoSCoW chart (1 hour)
- Final revisions to SPR1 document (2 hours, 30 minutes)
- SPR1 prototype candidate (50 minutes)

Tyler Deschamps:

- Review requirements for SPR2
- Develop beginning/working burn-down chart (1 hour)
- Gantt chart development (2 hours)
- Logging for Gantt/pert charts (1 hour)

Jordan Van Patten:

- Review requirements for SPR2
- V&V covering requirements and documentation (3 hours)

Topics Discussed

- Task management
- Overview of pert chart

1/24/2022

- Overview of Gantt chart
- Overview of work breakdown structure (WBS)
- Overview of high-level design document (rough draft)
- Use-case scenarios for high-level design
- V&V
- Final revisions to requirements spec
- Schedule next shareholder meeting
- Options for testing software

Obstacles Encountered

- Jaden found new solution for tracking tasks on team Trello board, but is still unable to edit tasks on his own list/card

Finished Items

- High-level design
- Fully viable work breakdown structure and MoSCoW charts
- Updated meeting logs to fit new template
- Scheduled next team/shareholder meeting for February 7, 2022

Unfinished Items

- Final requirements spec
- Use-cases for high-level design
- Create classes for GUI
- Finish developing pert/Gantt charts

Tasks Until Next Meeting

- Create classes for GUI
- Requirement spec
- Use-case scenarios for high-level design
- V&V
- Develop pert/Gantt chart

Notes

- At the start of SPR3, our team will start holding short meetings after class on Wednesdays