**Project Report #2**

**Team:** Codework Digital

**URL:** **thetalkmadesimple.com**

**Team Members:** Cat Schnelle, Chanse Stephens, Kevin Sims, Elijah Pemberton

**Group #:** 4

**Course:** CSCI-441-VA-Software Engineering

**Individual Contributions Breakdown**

**Report #2**

Chanse Stephens

* Interactive Diagrams
* Class Diagram
* Traceability Matrix
* Architectural Styles
* Persistent Data Storage
* Network Protocol
* Global Control Flow
* Hardware Requirements
* User Interface Design and Implementation
* Project Coordination and Progress Report
* Breakdown of Responsibilities

Cat Schnelle

* Table of Contents
* Project Management (Gantt chart + Trello board)
* Report Formatting
* UML Package Diagram
* Identifying subsystems
* Design of tests
* Plan of work
* Roadmap

Elijah Pemberton

* Design of tests

Kevin Sims

Table of Contents

pg. 1 - Cover page and Individual Contributions Breakdown

pg. 3 - Table of Contents

Part 1:

pg. 4 - Interaction Diagrams

pg. 7 - Project Management

Part 2:

Class Diagram and Interface Specification

* Class Diagram
* Data Types and Operation Signatures
* Traceability Matrix

System Architecture and System Design

* Architectural Styles
* Identifying Subsystems
* Mapping Subsystems to Hardware
* Persistent Data Storage
* Network Protocol
* Global Control Flow
* Hardware Requirements
* Project Management

Part 3:

User Interface Design and Implementation

Design of Tests

Project Management and Plan of Work

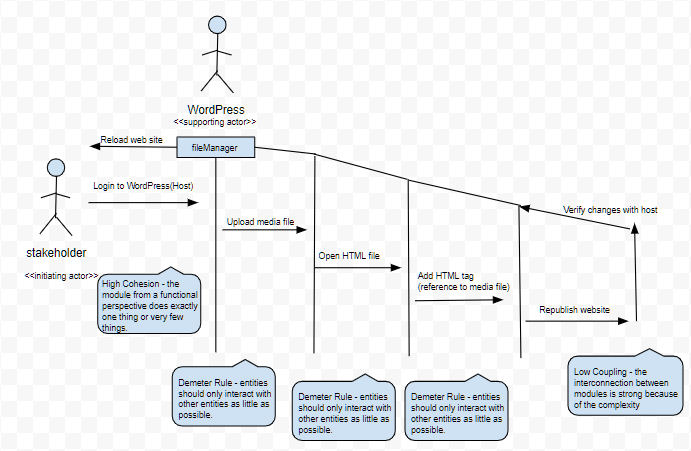
* Merging the Contributions from Individual Team Members
* Project Coordination and Progress Report
* Plan of Work
* Breakdown of Responsibilities

References

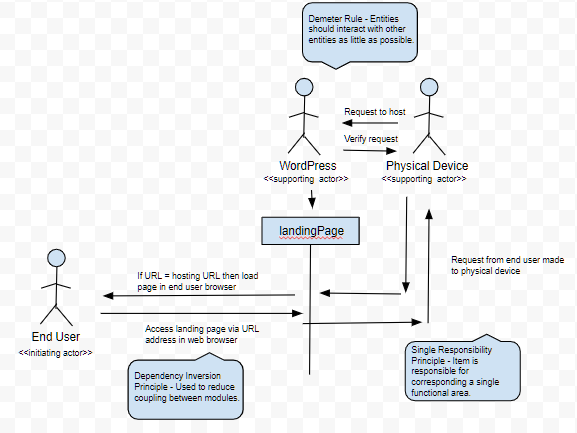
Part 1:

* **Interaction Diagrams**

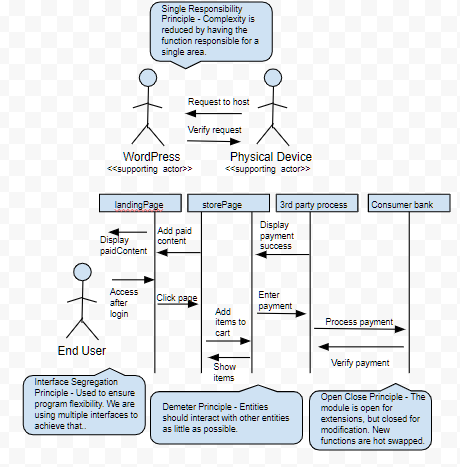
**UC-1**

****

**UC-5**

****

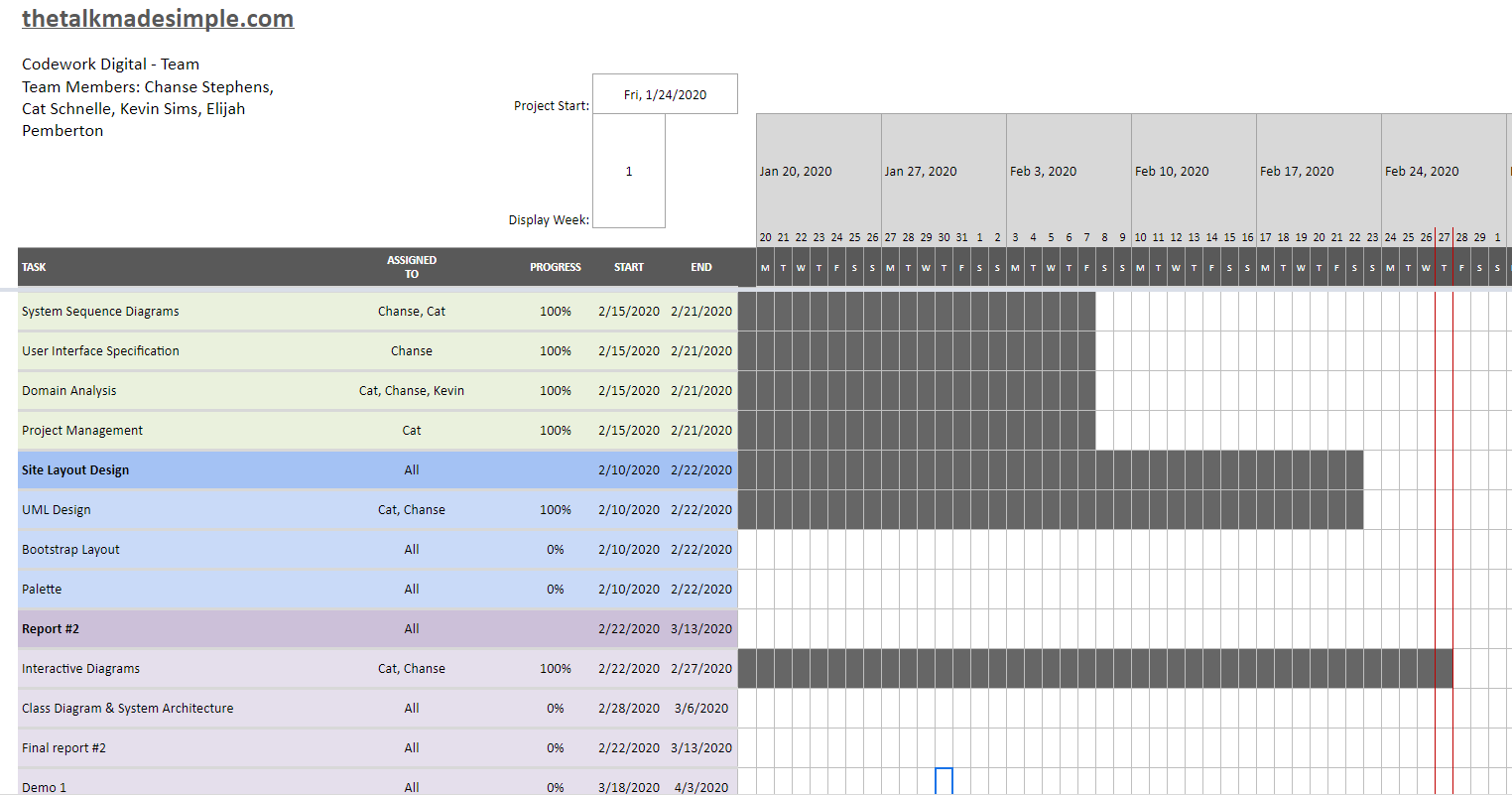
**UC-9**

****

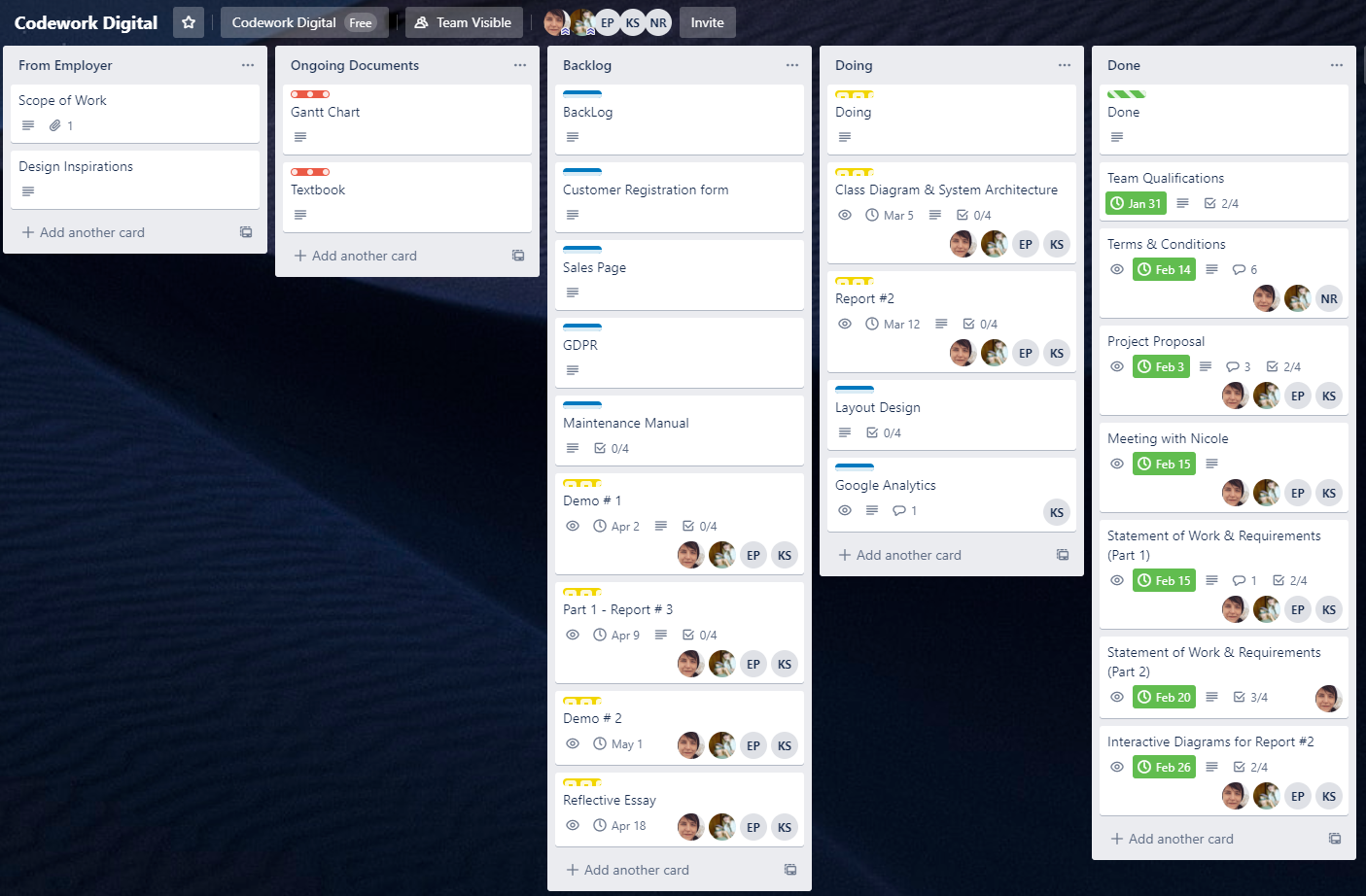
* **Project Management**

For part 1 of report #2; we have begun to take design input from our client in order to begin our bootstrap layout implementation next week for the landing page. This includes site layout features and color palette. We will be meeting with our client via phone for feedback regarding our initial implementation and making adjustments to the layout the following week. We have completed the interaction diagrams for part 1; using our sequence diagrams and elaborating on our employed design principles. Our goal is to begin part 2 of this report as outlined in our plan of work or Gantt chart below. We are active in utilizing Trello for our project roadmap; see an updated screenshot of our activity below.

Plan of Work:

****

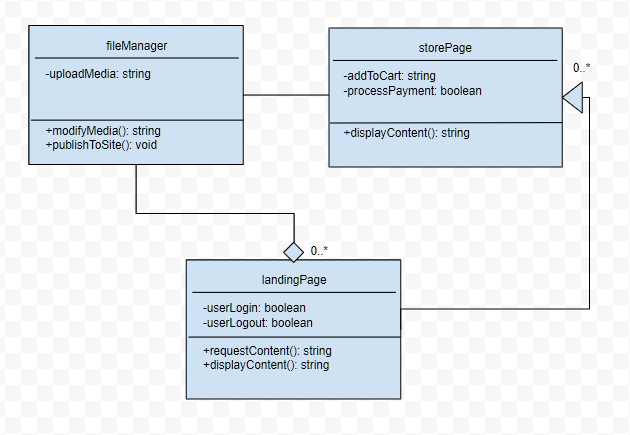
Roadmap:

****

Part 2:

Class Diagram and Interface Specification

* **Class Diagram**



* **Data Types and Operation Signatures**

***fileManager:***Class helps manage files on the WordPress site by allowing media to be uploaded to the site for various uses.

***uploadMedia*:** Attribute is private and is a string value. The value is correlated with the piece of media.

***modifyMedia*:** Operation is public and enables the user to modify the media to their liking. The media then returns a new string to update the old media with.

***publishToSite*:** Operation is public and is the piece that actually pushes the updated content to the site.

***landingPage:***Class allows users to view content and request content. This page also gives the ability for users to login and logout in order to utilize paid content.

***userLogin:*** Attribute is private and is a boolean value. This allows the user to login to the site.

***userLogout:***Attribute is private and is a boolean value. This allows the user to logout when finished accessing the site.

***requestContent:***Operation is public and lets the user request different versions of the landing page to be displayed.

***displayContent:*** Operation is public and is the piece that actually pushes the content to be displayed.

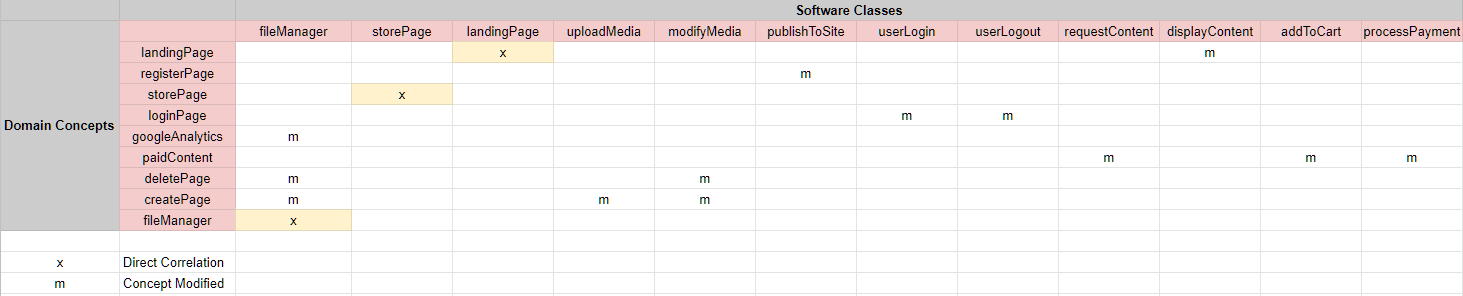
***storePage:*** Class deals with the store page. This page allows users to purchase content that’s locked behind a paywall.

***addToCart:***Attribute is private and is a string value. The value is added to the backend to keep track of what the user is wanting to purchase.

***processPayment:***Attribute is private and is a boolean value. This tells the system if the user has decided to purchase the content.

***displayContent:***Operation is public and is the piece that actually pushes the content to be displayed.

* **Traceability Matrix**



The fileManager class evolved from googleAnalytics, deletePage, and createPage. The class was modified to allow for greater control in a single class.

The uploadMedia attribute was created out of the createPage class from the Domain Model. The uploadMedia attribute is a component in the fileManager class.

The modifyMedia operation was modified from deletePage and createPage class from the Domain Model. The modifyMedia operation is also a component in the fileManager class.

publishToSite is a more straightforward version of the registerPage class from the Domain Model. This was created to make its use more clear.

userLogin is an attribute that was created out of the loginPage function from the domain model. This was created so that it could function in the landingPage class.

userLogout is an attribute that was created out of the loginPage function from the Domain Model. This was created so that it could function in the landingPage class.

requestContent is an operation that stems from the paidContent function from the Domain Model. This feature was modified so that it could be apart of other functions rather than just for paid content.

displayContent similar to requestContent, this operation was created so that it could be utilized in multiple instances instead of a singular operation. It was modified from the landingPage in the Domain Model.

addToCart is an attribute that was created out of the necessity of something to push along the paidContent operation from the Domain Model.

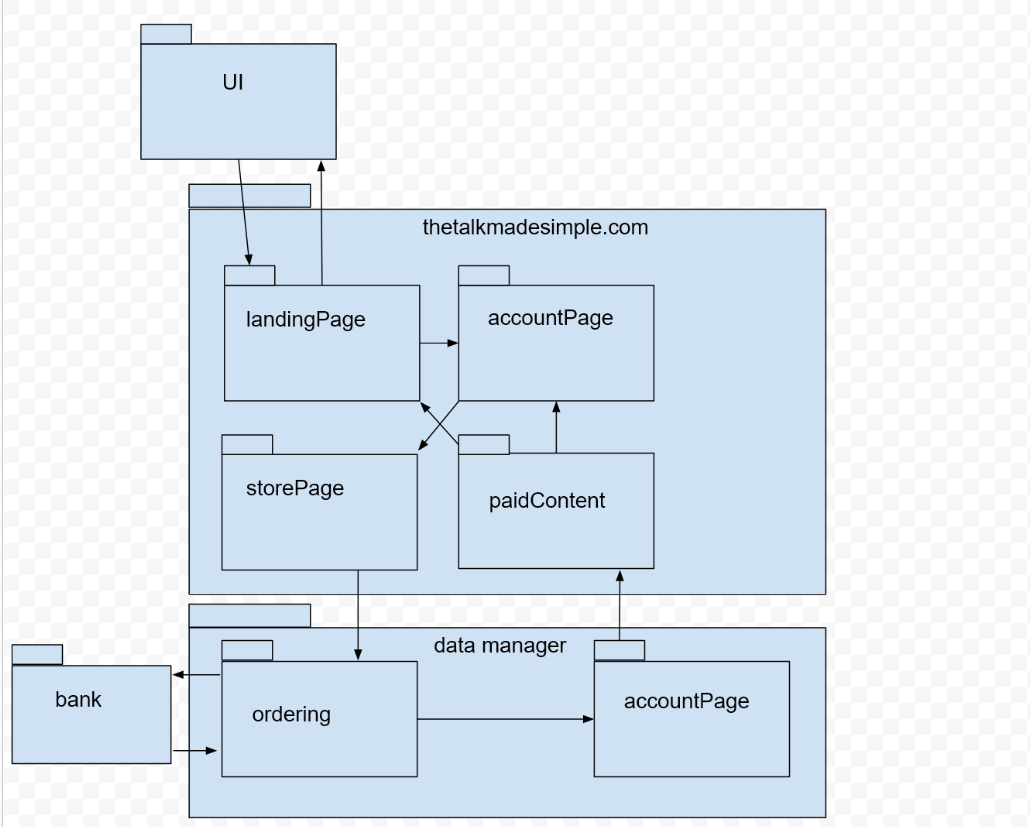
processPayment is another piece needed in order to fully realize the paidContent feature from the Domain Model.

System Architecture and System Design

* **Architectural Styles**

In our system which utilizes WordPress, the architectural style is closest to the Client-Server pattern. The pattern has two parties, the server and a single client or multiple clients. The server piece of the architecture which is WordPress provides services to multiple clients. The requests from clients are filtered through the server to retrieve the required end result. The server will continually listen for requests from the clients.

* **Identifying Subsystems**



* **Mapping Subsystems to Hardware**

The system really only needs a web server to function. However, to complete the program user’s must access it via a client workstation. The system is a website.

* **Persistent Data Storage**

The system will need to save some client data onto the web server. This data will be stored in the WordPress database. WordPress utilizes MySQL which is an open source relational database. Some of the data that will need to be saved are User Accounts, Content that’s been paid for, Analytics, Content Creation, and Notifications. WordPress is installed with 11 database tables by default. Those tables are…

1. wp\_posts
2. wp\_postmeta
3. wp\_options
4. wp\_users
5. wp\_usermeta
6. wp\_term\_taxonomy
7. wp\_terms
8. wp\_terms\_relationships
9. wp\_links
10. wp\_comments
11. wp\_commentmeta

* **Network Protocol**

Our system/site will utilize the HTTP and HTTPS protocols. These are the standard ports used to communicate with the web. HTTPS is used for security while HTTP is maintained for compatibility.

Mozilla describes HTTP and HTTPS as…

***HTTP*** *is a* [*protocol*](https://developer.mozilla.org/en-US/docs/Glossary/protocol) *which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is reconstructed from the different sub-documents fetched, for instance text, layout description, images, videos, scripts, and more.*

***HTTPS*** *(HTTP Secure) is an encrypted version of the* [*HTTP*](https://developer.mozilla.org/en-US/docs/Glossary/HTTP) *protocol. It usually uses* [*SSL*](https://developer.mozilla.org/en-US/docs/Glossary/SSL) *or* [*TLS*](https://developer.mozilla.org/en-US/docs/Glossary/TLS) *to encrypt all communication between a client and a server. This secure connection allows clients to safely exchange sensitive data with a server, for example for banking activities or online shopping.*

* **Global Control Flow**

Our website’s execution order is event-driven. Each user can choose which link to click on which will take them to various parts of the website. As for the time dependency our system is based on event-response. The system does not have a real concern for real time. When events are triggered the system will provide data. The WordPress content management system does not support multiple threads, so our website will not support it either.

* **Hardware Requirements**

Monitor Resolution: 1024 x 768 or higher

Minimum of 2 GB or RAM

An Internet Connection. Recommended speed 2Mbps or higher.

Keyboard and Mouse

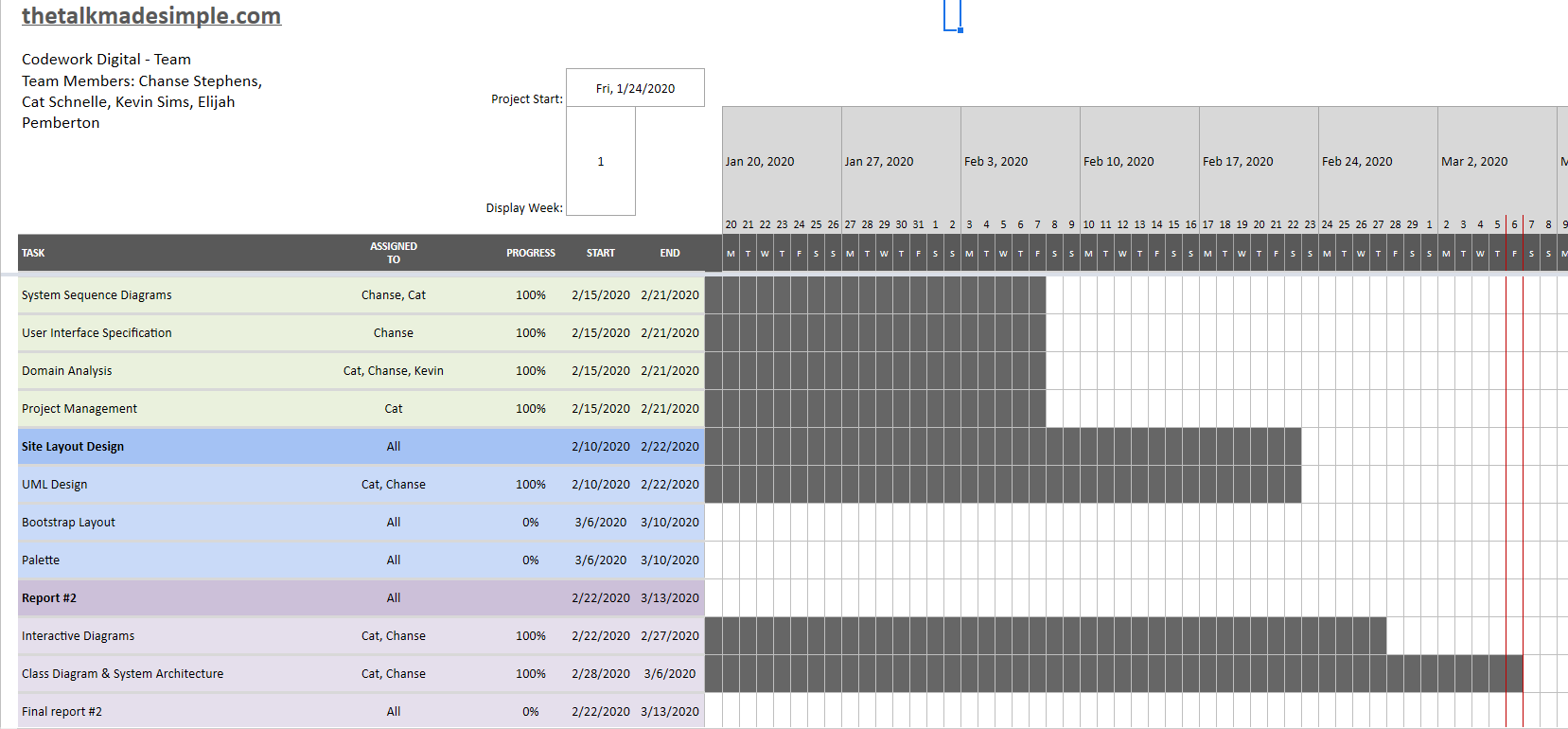
Software Requirements

Modern Web Browser such as Google Chrome, Mozilla Firefox, Edge, Safari

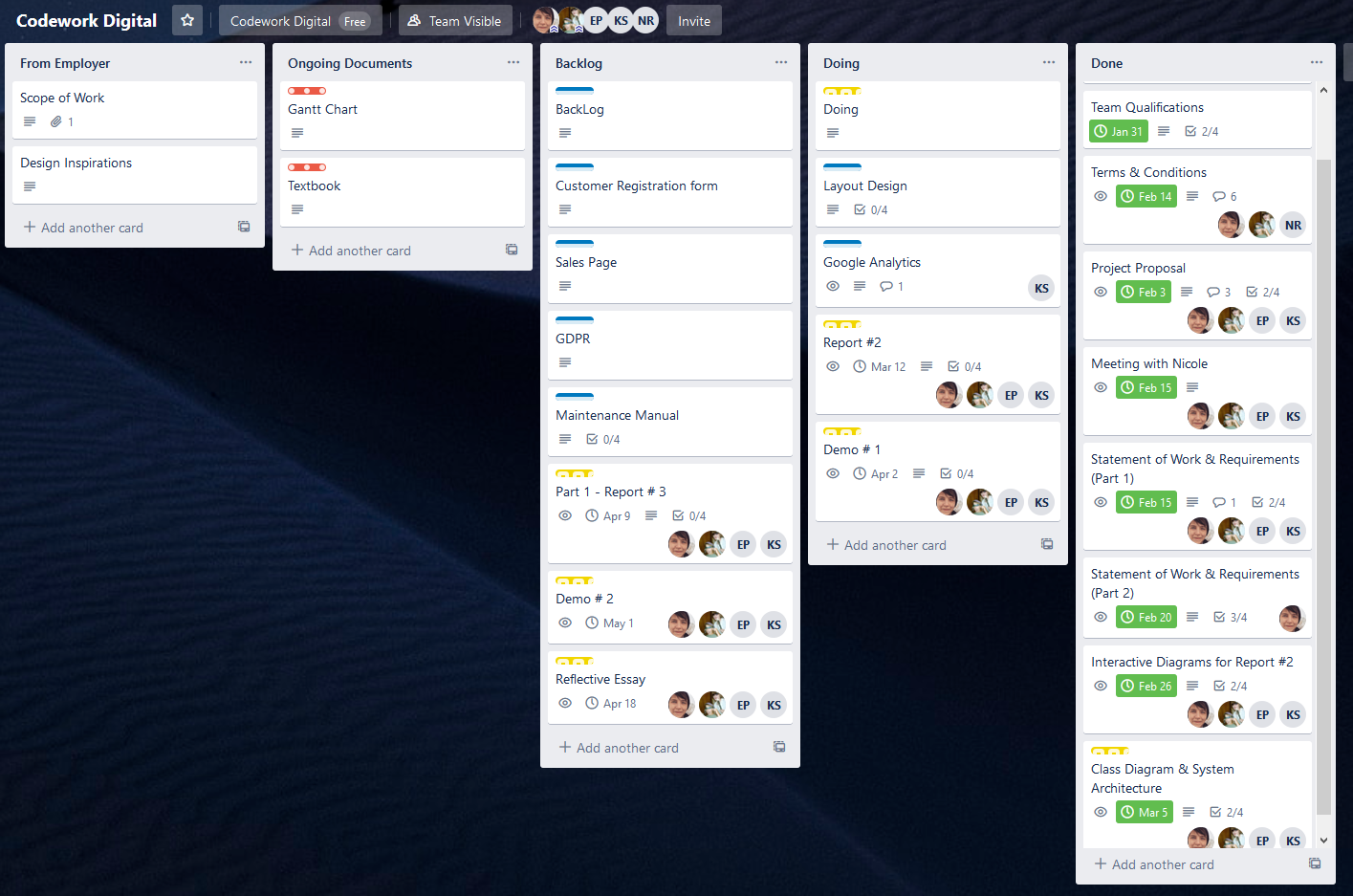
* **Project Management**

This week our team worked to finish part 2 of report 2; we are continuing to design the landing page for our first demo. We have decided to perform our weekly meeting with our client next Friday to walk through the initial building blocks of the demo in order to implement design feedback. Chanse and Cat have worked on completing our report as updated in the breakdown and will update the team on the next steps in the project. We are continuing to use Trello for our project roadmap and update our Gantt chart for our plan of work. Below these items have been updated to reflect this week and included:

Plan of Work



Roadmap



* **User Interface Design and Implementation**

As of now we are still modifying and tweaking the final design of the site will be probably until the end of the project. We had a good idea of what we wanted the site to look like in the beginning which gave us a good baseline. In order to keep the client happy and adapt with flow of the work we will be making design changes as the project progresses. One of the main changes thus far is the change in the look of the course layout. Previously, the course layout showed the course steps on the top of the page around the header area. Now the course steps are going to be shown on the right side of the page. This will give the customer more of a sense of progression as they move down the list of requirements to complete the lesson. Other changes are minor things like color changes, shape changes, and imagery. The main functionality of the design is in place which includes the upload features, course creation, payment, admin, etc.

* **Design of Tests**

Listed below is each case of our website’s core function that is expected to be working once the site has been launched:

1. Publishing content to the site via host (i.e Wordpress).
2. Content visibility on all pages.
3. Content visibility on paid account pages.
4. Creating user accounts.
5. Login with a user account.
6. Delete a user account.
7. Webpage can be navigated with links.
8. Webpage forms can be filled in and have default fields.
9. The website is consistent on both desktop and media platforms.
10. All media content on the website is visible on both desktop and media platforms.
11. All cookies are encrypted before being stored on the user’s machine.
12. The website encryption security verification is working in all browsers.
13. Free content is visible on both paid and free account types.
14. Paid content is only visible on paid account types.
15. Payment and banking information for users are not stored on the website.
16. Payment transactions are encrypted.

Listed below is each category of tests that will be performed on our website, these include; functionality tests, usability tests, interface tests, and compatibility tests. Below each category is the complete test coverage included in each test type.

**Functionality Tests**

**Links:**

1. Test all outgoing links on each web page.
2. Test all internal links on each web page.
3. Test to ensure there are not unreachable pages.
4. Test all links to external data hosts (i.e. email links, account creation links, payment links, sign in / sign out links)

**Forms:**

Test all forms on each webpage. (i.e. account creation form and payment information form.)

1. Test each validation field on forms.
2. Check that all default values in fields are correct.
3. Test that fields are correctly displaying incorrect inputs to the user.

**Cookies:**

Testing cookies by enabling and disabling cookies in all main browsers (i.e. mobile platform browsers, desktop browsers; Google Chrome, Firefox, Opera, Internet Explorer, Safari)

1. Check login sessions after ending sessions.
2. Check that cookies are encrypted before they are stored.

**HTML/CSS Validation:**

1. Check that website is searchable with all major search engines.
2. Validate HTML/CSS files using “W3C” markup validation service to check for errors.
3. Verify website encryption security service is working.

**Data:**

1. Verify that data entered into forms in account creation are accurately stored.
2. Verify that data stored in account creation are deleted upon request.
3. Verify that payment information entered in the validation field in forms is not stored after payment is completed.
4. Verify that payments are successfully completed.
5. Verify user access to paid/locked content is available once payment is recorded on account.
6. Verify that account status “paid/free” is accurately saved on each user account.
7. Verify that all accounts default to “free” account type.

**Useability Tests**

**Ease of Use:**

1. Verify that all elements on website are visible on all desktop and mobile web browsing platforms.
2. Verify that all elements are displayed in the correct location to layout mapping.
3. Verify that all links are displayed and visible.
4. Verify that website appearance is accurate to layout mapping.
5. Verify the website color scheme and check for accuracy in design requirements and allow for ease of use in various accessibility user cases. (i.e. limited use on complimentary colors for color blind accessibility, etc.)
6. Verify that all website links and media links are easily accessible on mobile and desktop platforms.
7. Verify that the website can be quickly and easily navigated.
8. Check that all website media content is in the correct location on all webpages.
9. Check that all website form fields are accurately displayed on the correct pages.
10. Check that all website content are the correct size on both mobile and desktop browsing platforms.
11. Check that all media links and navigation links are properly labelled and allow for voice navigation for users with sight accessibility requirements.
12. Check for any text and spelling errors in all text based media on each page.
13. Verify that the site is easy to navigate and read in multiple browser “dark modes”.

**Interface Tests**

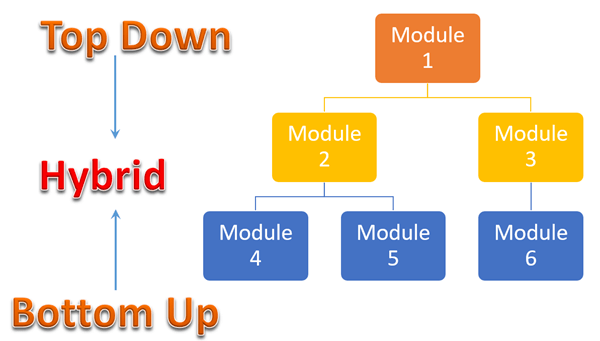
1. Verify that Google Analytics is implemented and functioning properly.
2. Verify that the payment system is functioning properly.
3. Verify that user accounts are defaulted and functioning.
4. Verify that user’s ability to create and delete their account is functioning.
5. Verify that account password security is functioning.
6. Verifying that incorrect validation fields are properly displaying error messages to the user.

**Compatibility Tests**

1. Verify website is consistent in all browsers on desktop and mobile platforms.
2. Verify that the website is compatible with all operating systems. (i.e Windows, Linux, Mac OS, Android, and IOS)
3. Verify that all media properly loads correctly and is properly autosized on all desktop and mobile platforms.
4. Verify that all forms are properly sized and visibly consistent on all mobile and desktop platforms.
5. Verify that any options to print site content is properly formatted on all desktop and mobile platforms.

**Integration testing**

For our integration testing we will be utilizing the incremental hybrid or “sandwich” approach.



In this strategy, testing is done by joining two or more modules that are logically related. Then the other related modules are added and tested for the proper functioning. The process continues until all the modules are joined and tested successfully. We will be using stub and drivers which are just dummy programs that test out our modules in order to test each individual one before connecting one module to another.

The advantages of the hybrid/sandwich strategy:

1. Allows parallel testing of various elements of the software system
2. Enables early testing of user interface components
3. Performs more coverage with the same stubs
4. It is a time saving process as several components are tested simultaneously

Upon adopting this test strategy; we will then begin to prepare our test data and cases. Using the interface and architecture design of our website we are able to map our critical modules for testing and map each use case that is logically related based on our test descriptions above.

* **Project Management and Plan of Work**
* Merging the Contributions from Individual Team Members

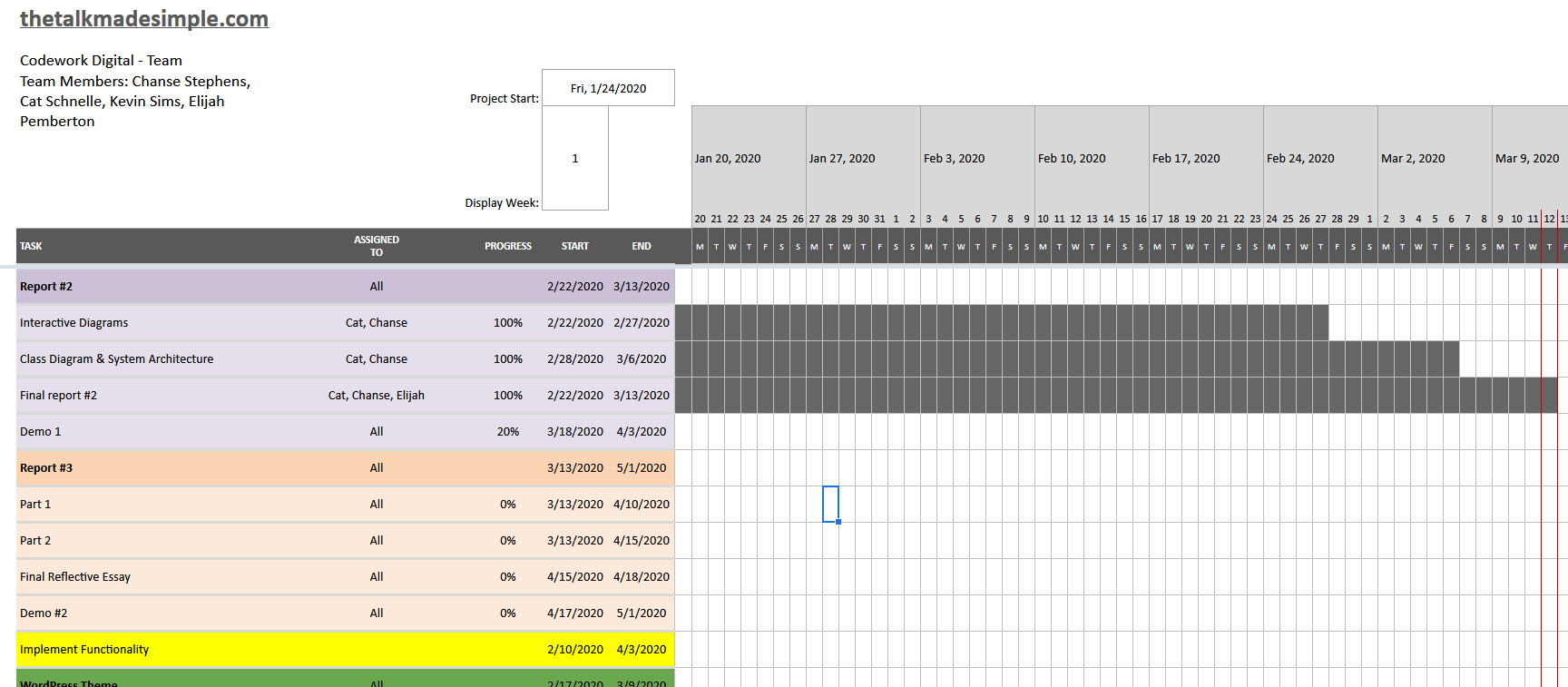
All pieces of the project are going to maintained in GitHub. The contributions are then merged on a single team members computer. That member then reviewed the files to ensure spacing and formatting are consistent. Then the member uploaded combined files into GitHub for the other members to view. The other members are responsible for reviewing files also in order to ensure consistency.

* Project Coordination and Progress Report

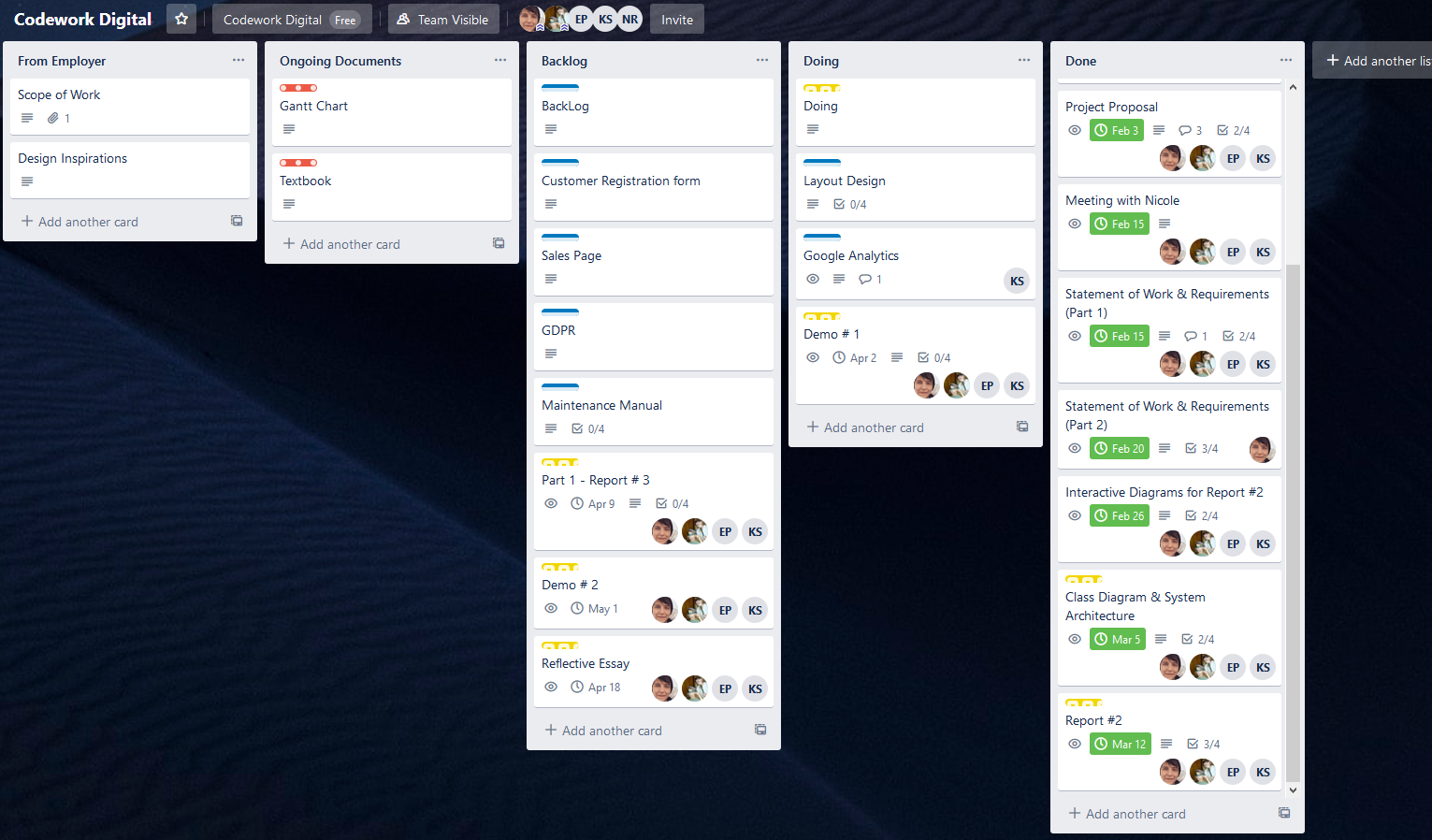
In regards to the use cases, the basic functionality of the sales page has been created. A customer is able to select a course to purchase and almost finish the transaction. Additionally, member accounts have been implemented. Users are able to login and view account details. Content creation is working as well, a lot of that piggy-backs off of the WordPress baseline functionality. Ease of use and content navigation is being actively developed. This portion of the project will be continually developed as the whole site is built. Features that still need to be implemented are site notifications to customers, training documentation, content presentation, and the mobile/responsive design of the site.

Other project management activities will include things like compiling pages of the WordPress site from other members of the team on GitHub. Once those pages are located in one place, we will then need to organize them in a folder to create the full WordPress theme that includes all use case functionality.

* Plan of Work



* Roadmap



* Breakdown of Responsibilities

Currently, for the most part we are breaking down our responsibilities through functionality. As of right now the responsibilities are as follows…

* Overall Design of Site: Every member has a say in the design, Chanse Stephens is leading the design and gets input from other members. Integration will be done by Chanse Stephens, but checked by other members of the team.
* Customer Accounts: Cat Schnelle is implementing, but receives help from other members of the team. This function includes the userLogin and userLogout operation and attribute. Integration will be done by Cat Schnelle, but checked by other members of the team.
* Customer Notifications: Elijiah Pemberton is implementing, but receives help from other members of the team. Integration will be done by Elijiah Pemberton, but checked by other members of the team.
* Learning/Sales: Chanse Stephens is implementing, but receives help from other members of the team. This function includes the storePage, processPayment, and storePage operations/attributes. Integration will be done by Chanse Stephens, but checked by other members of the team.
* Google Analytics: Kevin Sims is implementing, but receives help from other members of the team. Integration will be done by Kevin Sims, but checked by other members of the team.
* WordPress: All members of the team are contributing to the configuration of WordPres. This feature includes the fileManger, uploadMedia, modifyMedia, publishToSite, requestContent, displayContent, and landingPage operations/attributes. Integration testing will be done by all members.

**References**

**GitHub:** GitHub is a web-based version-control and collaboration platform for software developers. Git is used to store the source code for a project and track the complete history of all changes to that code. It allows developers to collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple developers.

Link: <https://help.github.com/en>

**W3C Validation Service:** This website service is a free to use HTML/CSS file upload tool that checks these files for various logical and syntax errors, automatically.

Link: <https://jigsaw.w3.org/>