# Milestone 4: Development Phase (Coding and Testing)

Implementation Plan

The implementation plan provides a detailed outline of activities necessary to ensure that delivery of the planned user stories for the milestone assignment. This course does not specify which planning delivery methodology or tools you are required to use in your implementation plan. It is recommended you seek advice from your mentor or the instructor. The following are some suggestions you can follow for your implementation plan.

1. Use a formal methodology, such Agile Scrum or Kanban.
2. Planning tools could include but not limited to Microsoft Excel or Word template, boards on monday.com, boards and reports on notion.so, scrum burn down templates, and more.
3. At a minimum your implementation plan should include the follow elements:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case or User Story | List of detailed Development Tasks | Estimate (hrs.) | Actual (hrs.) | Percent Complete |

Percent of User Stories complete for this iteration: 50%

Percent of User Stories complete for entire project: 40%

Estimated Time: 180 hrs.

Actual Time to Date: 113 hrs.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Title: CST451-Capstone Garden Watering System** | | | | | |  |  |  |
|  |  |  | **User Stories** | | |  |  |  |
| **ID** | **Feature** | **User Story** | **As a(n) <actor>** | **I would like to <description>** | **So that <outcome>** | **Priority** | **Sign-off Date** | **Modification Date** |
|  |  | Functional | Business Owner | Have a RESTful webpage for customers | I can grow my business | 1 | 29-Apr-22 | 5/15/2022 |
|  |  | Functional | Business Owner | Have a Login Page for Customers | I can deliver member content | 1 | 29-Apr-22 | 5/5/2022 |
|  |  | Functional | Business Owner | Have a Registration Page | Users can login into member content | 1 | 29-Apr-22 | 5/5/2022 |
|  |  | Functional | Business Owner | Track User information | I can have useful analytics provided to me | 1 | 29-Apr-22 | 5/10/2022 |
|  |  | Functional | Business Owner | have branded pages with logos and business colors | customers will remember my business | 2 | 29-Apr-22 | 5/10/2022 |
|  |  | Functional | User | be able to purchase package deals | I can save money and have extra benefits offered to me | 3 | 29-Apr-22 |  |
|  |  | Non-Functional | User | to have an automated subscription service | do not have to remember to purchase each month | 3 | 29-Apr-22 |  |
|  |  | Functional | Business Owner | To have multiple payment options | so that customers are not limited on how they pay | 3 | 29-Apr-22 |  |
|  |  | Non-Functional | Developer | Have a third party payment service | my company is not liable for regulations regarding payment information | 3 | 29-Apr-22 |  |
|  |  | Functional | User | Have a list of Predesignated timers for watering common plants | I do not have to determine best watering practices | 1 | 30-Apr-22 | in Progress |
|  |  | Functional | User | have a simple interface without much input | I can easily set up the system and understand what it is set to do | 2 | 30-Apr-22 | in Progress |
|  |  | Functional | Business Owner | use a third party security service | I can have up to date security measures without nessesity for re-deployment | 2 | 30-Apr-22 | 5/10/2022 |
|  |  | Non-Functional | Business Owner | have automated data backups | I do not lose customer and system data relevent to the business | 1 | 30-Apr-22 | 5/11/2022 |
|  |  | Functional | Developer | automated unit and system testing built in | the system will scale and test functionality at each deployment | 1 | 30-Apr-22 | in Progress |

*Table 1: Project User Stories*

*Chart

Description automatically generated*

*Image 1: Week 1 Burn Down Chart*

*Chart

Description automatically generated*

*Image 2: Week 2 Burndown Chart*

*Chart

Description automatically generated*

*Image 3: Week 3 Burndown Chart*

*Chart, line chart

Description automatically generated*

*Image 4: Week 4 Burndown Chart*

Source Code Listing

While this project began as a proposed MVC application, during the build an N-Layer type structure became noticeably necessary in order to include the necessary third-party integrations. Pictured below is the main “root”/“src” location with the main divisional breakdown, followed by each divisional breakdown. Source code can also be viewed on GitHub at the following link. <https://github.com/smoncavage/CST451_Capstone.git> This is a very large program with many interwoven pieces. Some of which listed below are deprecated and will be removed for the final production version.

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 5: Main Source Folder with N-Layers*

In Image 5 above, the main folder consists of the higher divisional folders along with the necessary connection files for Composer, GitHub, a main site index page which re-directs to actual home page and a backup of the necessary Raspberry Pi files.

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 6: Business Service Layer*

In the Business Service Layer we have the necessary models for each object listed below in Image 7. There also exists the necessary service layer for connecting each service to each of the models.

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 7: Models in Business Layer*

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 8: Data Connections Layer*

The Data layer connects to the Business Layer via Handlers which are shown below in Image 13. The main “db” file is the connection file necessary for MySQL database connection.

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 9: User Interface Layer*

The user Interface layer consists of the bulk of this project since it is a web-based application. A majority of the “css” folder consists of the third-party front-end integration files. Of which most of these are deprecated in this project and will be removed in further iterations of development. The pictured “javascript” and “pictures” folders are where the subsequent files will reside once the third-party structure is removed.

*Graphical user interface, text

Description automatically generated*

*Image 10: CSS files and Third-Party HTML files imported from Front-End Integration*

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 11: Third party Front-End Folders and File Structure*

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 12: Third party Front-End JavaScript Files*

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 13: UI Layer Handler Files*

The “handlers” folder holds the integration handlers between the Business and Data Layers for displaying the different associated page views with content.

*A screenshot of a computer screen

Description automatically generated with medium confidence*

*Image 14: Main User Interface Layer Views*

The “views” folder holds the bulk of the application files as they are directly responsible for end user navigation of the entire web application. The pictured “login” folder is deprecated and will be removed in further iterations of development.

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 15: Utility Layer for Miscellaneous Files*

A Utility “layer” exists for miscellaneous files which do not conform to the other structures. This layer holds the session validation and logging files necessary for proper testing and security. Other files will be scrubbed in further development, but are likely to be considered deprecated files to be removed.

*A screenshot of a computer

Description automatically generated with medium confidence*

*Image 15: All Associated Third-Party Integrations for Web Application*

The “vendor” folder contains all of the associated third-party files necessary for proper integration into the application and are kept up to date via the “composer-json” file in the main source folder. These files are kept up to date via composer in the CI/CD pipeline.

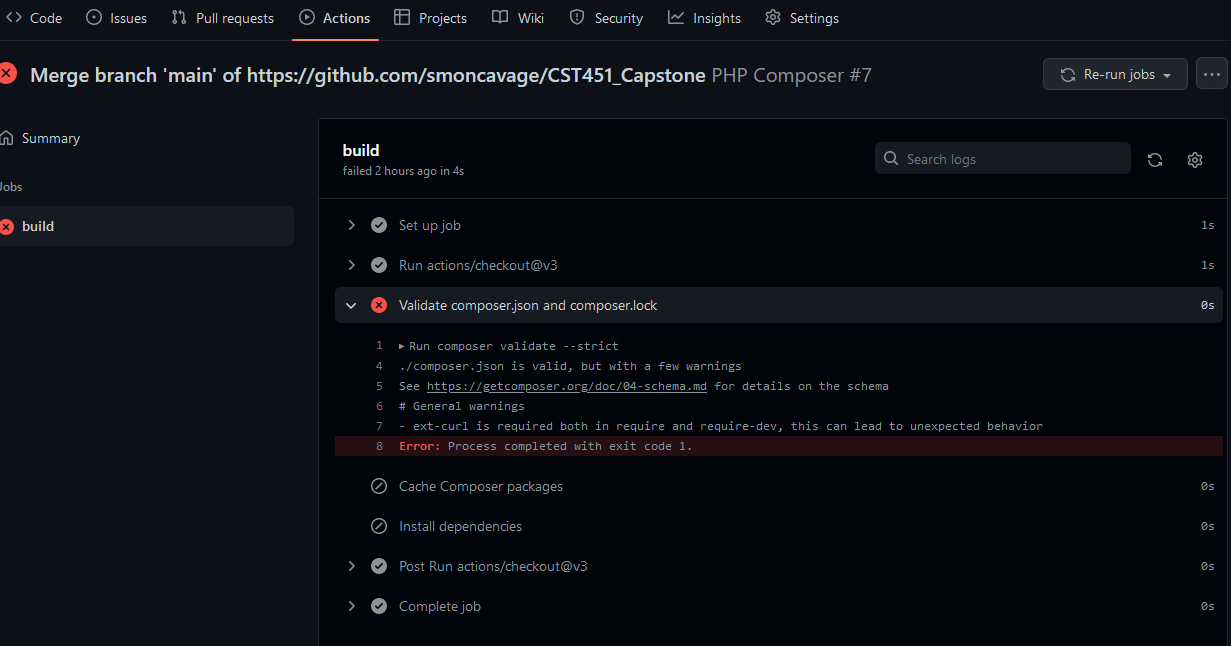
Test Plan and Test Cases

The only current testing being conducted in pipeline is security testing by third-party GitGuardian and PHP Composer file validation for up to date dependencies by GitHub. Some current Open Items are pictured below. Other development testing is being conducted in a local development environment for data structure unit testing to ensure proper data validation. Formal testing will be devised and built in the coming weeks.

Graphical user interface, text, application, email

Description automatically generated

*Image 16: GitGuardian Open Items*

**

*Image 18: GitHub Open Item*

Application Demonstration

Demonstration screencast can be viewed at the link below.

<https://screenrec.com/share/Ri05fQaOEG>