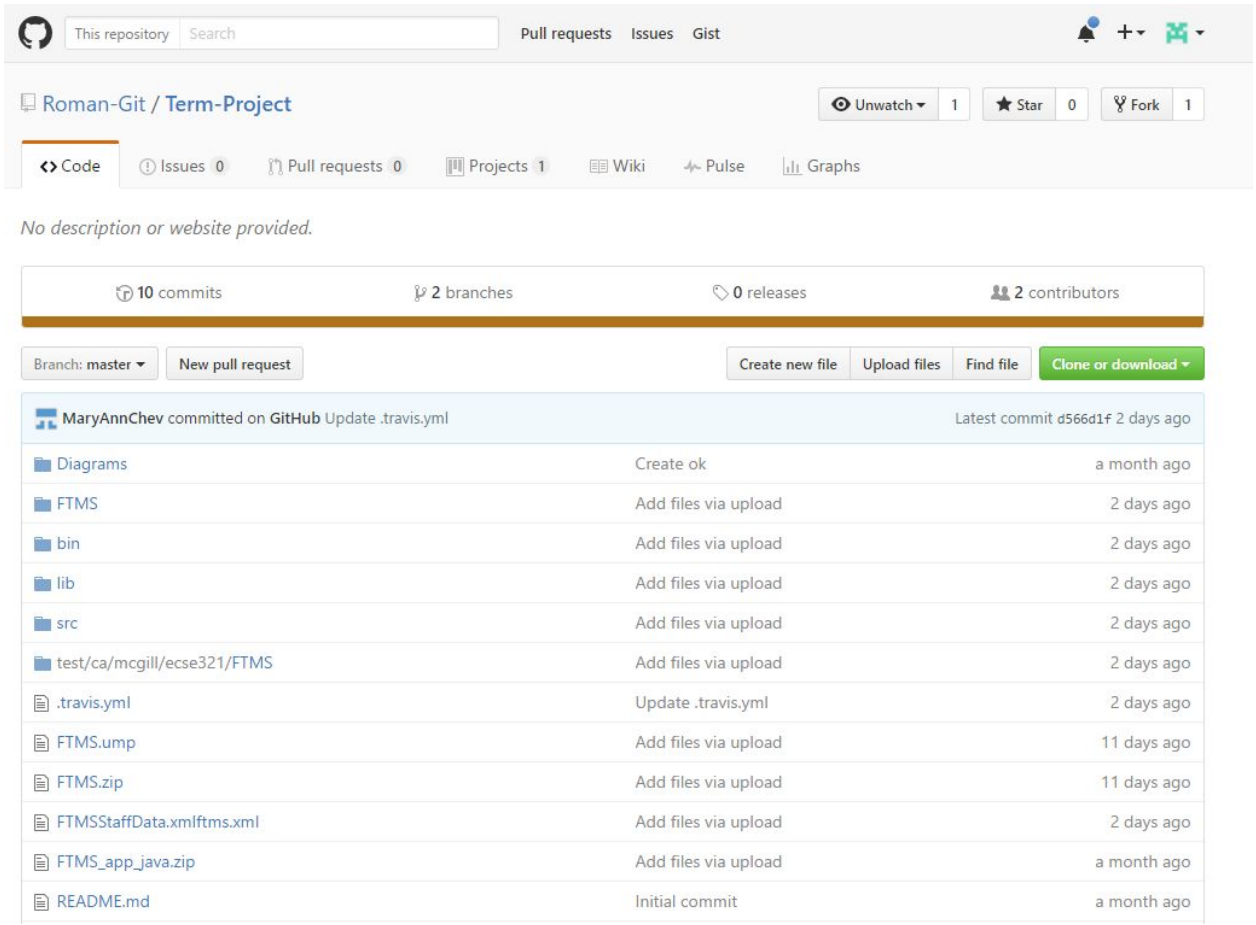
Release Pipeline Plan

Group 11 - Food Truck Manager Application - Deliverable 4

Integration - “GitHub”

Each of our project folders for each platform is stored via a source control system. For this project we are using GitHub as our version control tool. GitHub provides access to all members of the team. Every time a person commits specific files, we can see changes made to those files. If a change made during this stage is incompatible, we can always revert to old builds of our programs, making for a suitable back up system if something is not working as intended. GitHub is a service that is widely used by programmers working in teams and works for any kind of programming file or platform. For this reason we decided to use GitHub as our main Integration tool for all of our platforms (Web/Mobile/Desktop).

Figure #1: Snapshot of our Project GitHub Page for the Desktop Application



Source control software uses specific vocabulary to manage any actions done to the project. Below are a few example of key vocabulary necessary to use GitHub as our version control system.

Commit refers to accepting changed files into to a project.

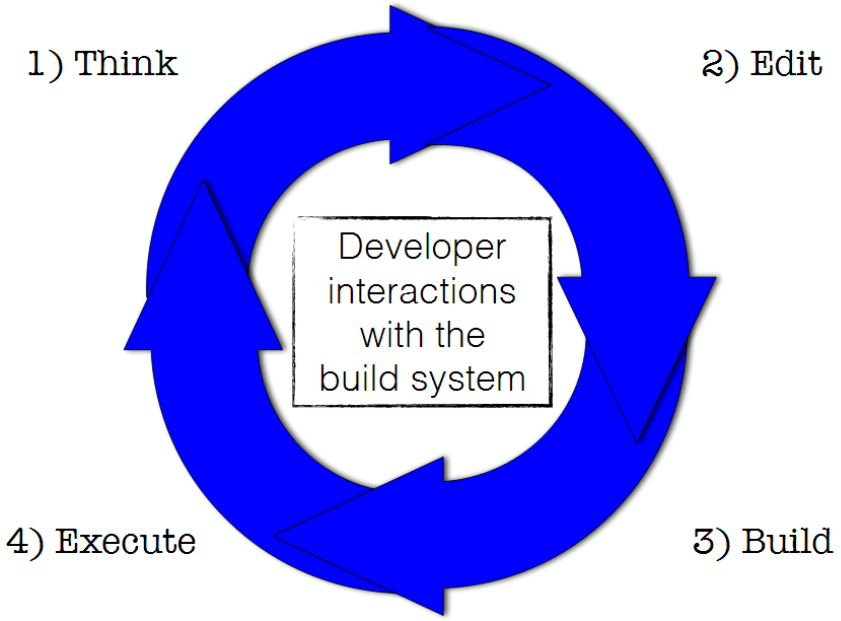
Push refers to uploading all of the changes and the metadata about changes that were committed on a local system.

Pull refers to updating files from a remote copy. When a contributor would like to modify files and they are not the owner, the proposed change is submitted as a *pull request*. If the pull request is accepted by the owner, the changed files are merged with the project repository.

Build

For our application, we will execute the build after the integration stage separately for each platform. This is because each application is unique in purpose, which means the end result requires different tools to complete the build. The build phase will be a continuous integration system for each platform. We will abide by the diagram shown in our ECSE 321 class below, where the steps Think, Edit, Build, Execute are treated as a cycle that we will continuously revisit until our product on each of our platforms are ready for deployment.

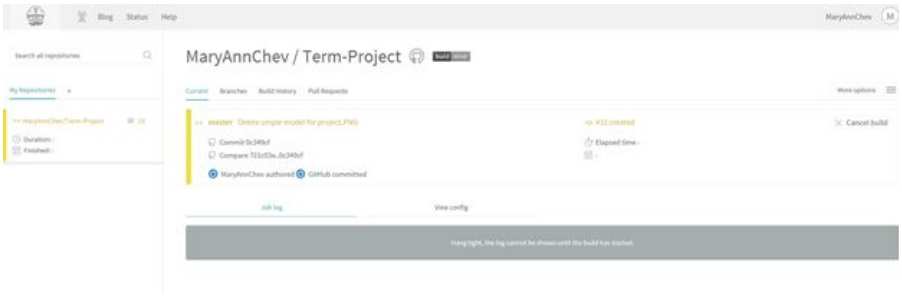
Figure #2: Continuous Integration Build System



Desktop

The desktop application will use a Web-hook which calls the source control system and builds the project. The web service we will use is Travis-CI, which is not only a popular tool for developers, but it is easy to use and flexible with different operating systems.

Figure #3: Travis-CI usage with our Desktop Application

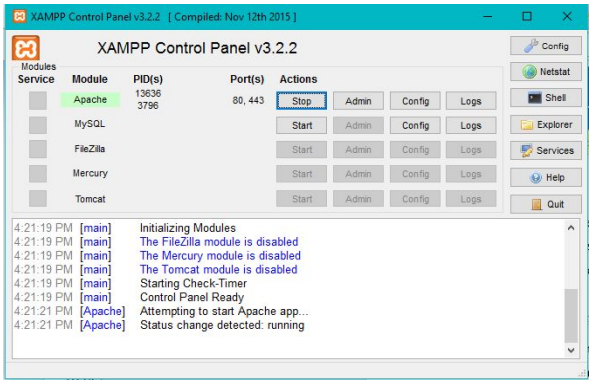


The first step is to link the repository to the Travis-CI website. From here we build the project which will output a jar executable that packages the content of our repository project all-in-one. This tool speeds up the building process and makes it easy for us to manage multiple builds if we find bugs that occur while testing.

Web

Since our web application is hosted on a server. We do not need to build the file in order to deploy it. Once integration stage is done, our plan is to run an Apache Server through XAMPP to host the website for our Food Truck Management Application. XAMPP is our go-to tool since it is a free tool and allows us to host our small sized project locally in order to test it amongst ourselves. For the final submission we hope to be able to have a server to host so that a simple click of a web address will allow the user to reach the web application. In essence, the web address is the “executable” build file for the web server.

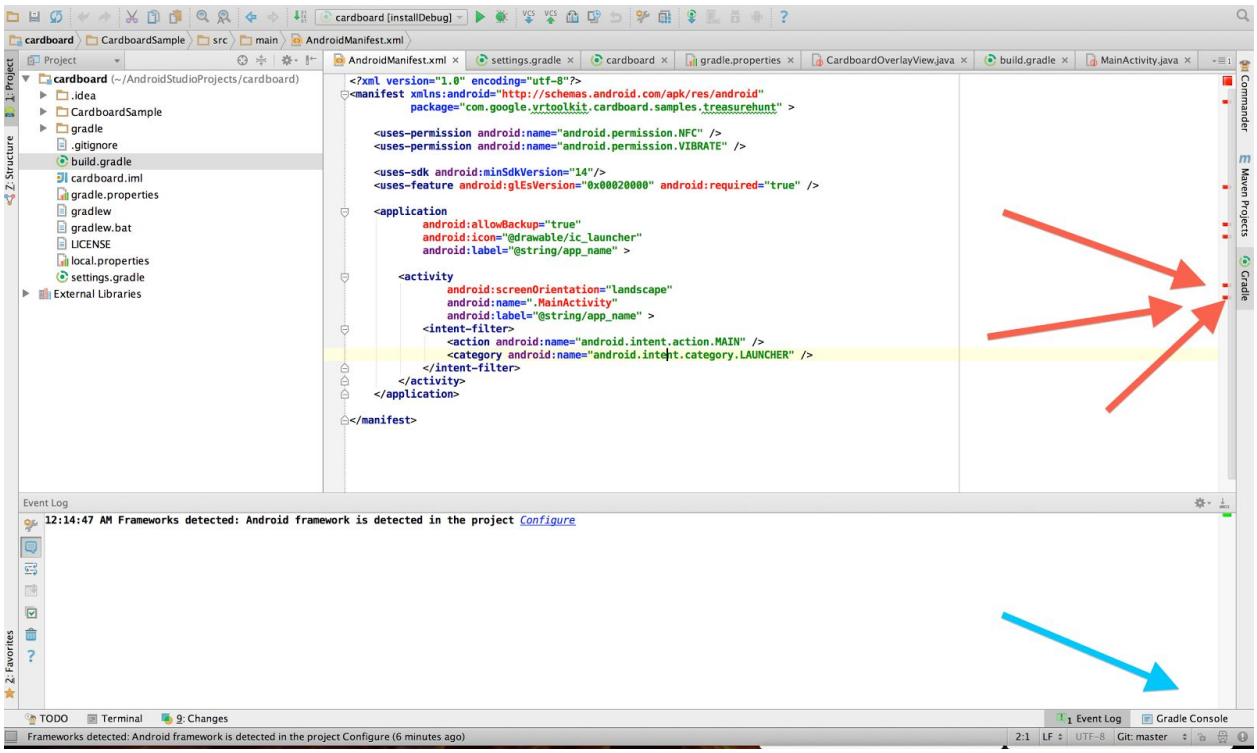
Figure #4: Example usage of XAMPP to host Apache server locally



Mobile

For the mobile application, we will build our project similar to the Java project. We will use Travis-CI as the webhook system. Travis-CI will use gradle to create a build file for us. Gradle is a default tool in Android studio, which makes Travis-CI much easier to use. Since it is flexible for android, if we run into bugs after a build file is created, it will be much easier to fix directly in Android studio to recreate and test a new build file.

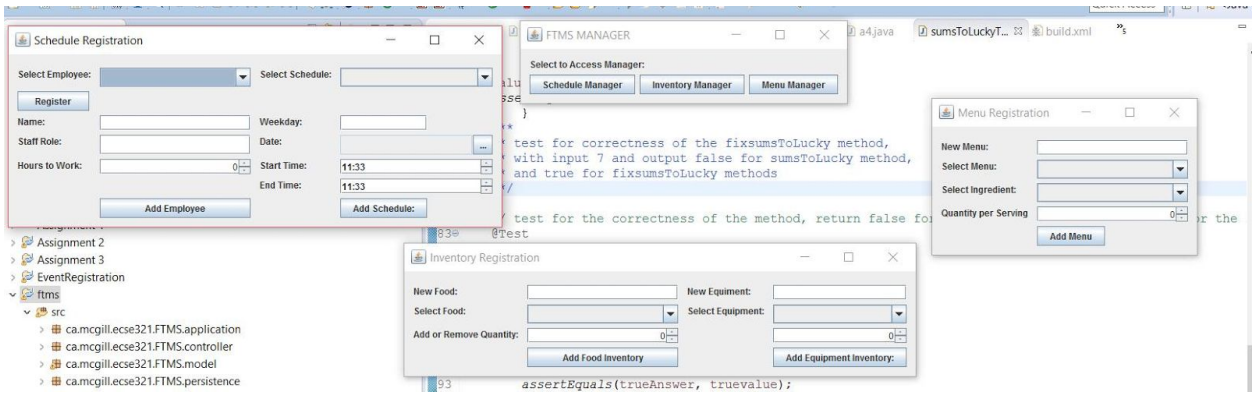
Figure #5: Example of gradle plugin accessible through Android Studio



Deployment

After the project is built, tested and re-built again, the deployment stage of the release pipeline is conducted. For each of our applications we will deploy our code via our source control system GitHub, this is the main way we will allow users to access and install our application. However, each platform may have other ways to access the program. For example, Android studio has its own tools that we can use for deployment. It can either deploy to the emulator with an android virtual device or it can deploy to a physical device. This is helpful if we need to test new features that we may implemented in a future build as well as any specific bugs relating to different phone models that need to be addressed.

Figure #6: Running our built jar file example for Desktop application



The project flows through this release pipeline plan in the following manner. Programmers are constantly making changes and improvements to the code. These changes are pushed to the Github repository frequently, with specific names and comments to keep track of each commit. When a change is committed, Travis CI will begin automatically building the changes. Afterwards, the tests written for the program are run ensuring that the latest commit does not hinder the program from functioning. If any of the tests fail, the build is unsuccessful and it is advisable to return to the previous commit. If all of the tests pass, then the deployment stage is performed. As the build file has been created, it is still possible to run this file to view the full program running as it would for the client. This will ensure that we will have easy access in order to test new features for future builds as well as for bug fixing in the current build.

Updated Work Plan - Nov 21, 2016 Food Truck Management App Group #11

Iteration #1 - Completed on October 28th, 2016

The weekend before our due date for deliverable 2 we completed our 1st iteration of our Food Truck management app. This included having a working view and functionality for adding an employee, creating a workday schedule, as well as registering employees to a workday schedule on each of our planned platforms (desktop, mobile, web). We also included a decisions section for our group meeting on October 27th, 2016.

- **Requirements completed:**
    - **Register an employee**
    - **Create a workday schedule**
    - **Register employees to a workday schedule**
- **Amount of hours used:**
    - **Estimated Hours: 30 Hours combined**
    - **Total hours: 24 Hours**
- **Decisions (Meeting - Oct 27, 2016):**
    - **Finish implementation on Desktop and PHP before working on Android since it requires completed Java code to implement view.**
    - **Error handling should be added as a general requirement for each requirement listed.**

Iteration #2 - Completed on November 14th, 2016

Before the due date for Deliverable 3, we designed, completed and tested the inventory handling and cashier manager portion of our application on all platforms on the Desktop and Web platforms for our application. This included completing the functionality to add an inventory item, to edit existing inventory items, to create a menu, to edit an existing menu, and to manage the popularity of purchases. We were unable to complete the android portion as we ran into issues integrating and displaying the Java code with the view.

- **Requirements to complete:**
    - **Add and edit inventory items**
    - **Add and edit a menu**
    - **List most popular purchases**
- **Amount of hours used:**
    - **Estimated Hours: 45 Hours combined**
    - **Total hours: 40 Hours**
- **Decisions (Meeting - November 11, 2016)**
    - **Finish implementation of desktop application before proceeding with android, since the mobile application requires the Java file from the desktop application**
    - **Make sure PHP and Java code is tested and working for both managers by the Iteration #3 due date.**

Iteration #3 - Completed on November 20th, 2016

For iteration 3, we set to design, complete, and test the remaining portion of our application. For the desktop application we have a working build that has been tested and rebuilt to satisfy our requirements. However, Android and PHP we have fallen behind due to

the lack of group members and contribution. Since we have now merged with another group of two, our focus for the last week until Deployment is to complete a working build for both Android and PHP. We will also try to implement any requirements we may have missed or thought of along the way to improve each of our platforms. We must also prepare the presentation for Deliverable 5 that includes performing live demo of our project.

- **Requirements completed:**
    - **Working build for Java Desktop Application**
    - **Android inventory manager**
    - **PHP inventory manager**
- **Amount of hours used:**
    - **Estimated Hours: 15 Hours combined**
    - **Total hours: 45 Hours**

Project Completion - Due by December 1st, 2016

Since we will test the sub-portions of our app between Iteration #1 - #3, during this phase we will test and debug our application as a whole. We will make sure that each function handles errors with any common scenario. On the project completion date, our application will be ready for deployment, ready for a live demo, and ready to present to the class.

- **Amount of hours required:**
    - **~12 Hours combined**
    - **~4 Hours per group member**
    -
- **Total Project Time:**
    - **~132 Hours combined**
    - **~26.4 Hours per group member**