U.S. Department of Education  
Federal Student Aid



New Joiner Projects

Version # 1.9.5

08/04/2022

Document Version Control

| Ver | Date | Author | Description |
| --- | --- | --- | --- |
| 1.0 | *12/5/2018* | *Peter George* | *Initial Draft* |
| *1.1* | *12/5/2018* | *Jacob Peterson* | *Added Automation Phase* |
| *1.2* | *12/10/2018* | *Peter George* | *Update Pre-Clearance Phase to incorporate Spring Boot 2.x in task 3+* |
| *1.3* | *01/23/2019* | *Antonio Perez* | *Added Pluralsight registration instructions* |
| *1.4* | *2/15/2019* | *Peter George* | *Added Pluralsight Required/Recommended courses* |
| *1.5* | *2/25/2019* | *Jacob Peterson* | *Added Unit Testing Task* |
| *1.6* | *4/15/2019* | *Jacob Peterson* | *Updated versions, added new post-phase shell* |
| *1.7* | *5/1/19* | *Joseph Slagle* | *Added Helpful Links to Preclearance Tasks* |
| *1.8* | *5/8/2019* | *Jacob Peterson* | *Updated DCC Auto post-phase* |
| *1.8.1* | *5/15/2019* | *Aaron Alcoser* | *Added Helpful Links Clearance for Pre-Clearance Task 5* |
| *1.8.2* | *9/12/2019* | *Nathan Kull* | *Added Asynchronous submission to Task 1* |
| *1.9.0* | *10/18/2019* | *Tucker Belton* | *Added Task 0* |
| *1.9.1* | *10/24/2019* | *Tucker Belton* | *Rewrite of Pre-Clearance Phase* |
| *1.9.2* | *11/21/2019* | *Tucker Belton* | *Updated requirements* |
| *1.9.3* | *11/25/2019* | *Bernard Siegel* | *Added suggested GitHub tutorial – to be reviewed* |
| *1.9.4* | *12/13/2021* | *Matthew Dowell* | *Task 0 SCM requirements, removed extraneous requirements.* |
| *1.9.5* | *08/04/2022* | *Per Van Dyke* | *Clarified Compatibility of Spring/Hibernate for task 2.* |
| *1.9.6* | *11/22/2023* | *Gloria Lee* | *Updated technology stack and relevant project requirements* |

Table of Contents

[Document Version Control 2](#_Toc92880467)

[1. Executive Summary 4](#_Toc92880468)

[1.1 Background & Purpose 4](#_Toc92880469)

[1.2 References and Related Documents 4](#_Toc92880470)

[2. Pre-Clearance Phase 6](#_Toc92880471)

[2.1 Task 0 – Application Concept and Source Control 6](#_Toc92880472)

[2.2 Task 1 – Develop a new Web Application 7](#_Toc92880473)

[2.3 Task 2 – Persistence Update to Web Application 9](#_Toc92880474)

[2.4 Task 3 – Implement Web Services (REST) for Web Application 10](#_Toc92880475)

[2.5 Task 4 – Unit Testing and Web Automation Testing 12](#_Toc92880476)

[2.6 Task 5 – Transition Web Application to Angular 13](#_Toc92880477)

[2.7 Task 6 – Additional Requirements (Optional) 14](#_Toc92880478)

[3. Post-Clearance Phase - Client Web Applications 15](#_Toc92880479)

[3.1 Task 1 – Local Workspace for Client Web Applications 15](#_Toc92880480)

[3.2 Task 2 Access – Simulated Change Request for Client Web Applications 16](#_Toc92880481)

[3.3 Task 3 Access – Repair Client Web Applications 17](#_Toc92880482)

[4. Post-Clearance Phase – TIVOD Testing Tools 18](#_Toc92880483)

[4.1 Task 1 – Local Workspace for Testing Tools 18](#_Toc92880484)

[4.2 Task 2 Access – Simulated Change Request for TIVOD Testing Tools 19](#_Toc92880485)

[4.3 Task 3 Access – Repair TIVOD Testing Tools 20](#_Toc92880486)

[5. Post-Clearance Phase – DCC Testing Tools 21](#_Toc92880487)

[5.1 Task 1 – Learn SeeTest 21](#_Toc92880488)

[5.2 Task 2 – Local Workspace for DCC Testing Tools 22](#_Toc92880489)

[5.3 Task 3 – Sample Flow Creation 23](#_Toc92880490)

# Executive Summary

## Background & Purpose

The Safety and Civilian Services (S&CS) Portfolio shall provide an orientation and aptitude project for all accounts with personnel from the Custom Advanced Technology and Architecture (AT&A) capability. This project will provide new team members an opportunity to perform guided and applicable training on the technologies employed by their respective accounts. This project also allows development managers an opportunity to assess aptitude, work ethic, and capability of new team members.

This project is divided into two (2) phases, pre- and post- clearance approval. There are three (3) options for the post- clearance phase, depending if the new team member will be joining the COD Web team, TIVOD Testing Tool Team, or DCC Testing Tool Team.

The pre-clearance phase will cover an eight (8) week period, 320 business hours and will cover the design and development of a new web application (webapp) from a simple project to a full-stack experience by its completion.

The post-clearance phase will cover approximately three-and-a-half (3.5) week period, 140 business hours and will cover the familiarization to the appropriate applications. This phase will also introduce change request implementation as well as troubleshooting and defect repair. For COD Web Team hires, the appropriate applications include the FSA webapps and supporting projects for StudentLoans.gov, COD Web, and Electronic Campus-Based (eCB). For TIVOD Testing Tool Team hires, the appropriate applications include the Automation Executable JAR, the Automation WebApp, and the Midrange Interactive Testing Environment (MITE). For DCC Testing Tool Team hires, the appropriate application is SeeTest Automation.

The culmination of this project should provide new team members with a level of confidence in the technologies employed by their respective accounts.

**DISCLAMER: Do not begin working on the New Joiner Project until after you have submitted your Eqip and completed all the required Ethics and Compliance training.**

## References and Related Documents

All reference and related documents are located on the COD Web and Testing Tool teams’ Confluence sites.

# Pre-Clearance Phase

## Task 0 – Application Concept and Source Control

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Read “New Joiner Project Documentation” documentation  Create a proposal for a new personal Web Application (webapp) |
| **Tech Stack** | * Git: <https://github.com/> * git-scm: Download from Software Center |
| **Conditions** | Read required documentation. All in-progress development work should be committed to a “dev” branch. Push a “review” and “dev” branch to github, for each project task, a pull request URL (“review” <- “dev”), will be created for each task review. |
| **Requirements** | 1. Come up with a name for proposed application 2. Web application should consist of a few web pages 3. Create a public Git Repository using your *accenturefederal.com* email 4. Create a main, review, and dev branch(s) |
| **Helpful Links** | |  |  | | --- | --- | | [Udemy Course on Github](http://www.udemy.com/course/git-started-with-github) | Getting started with GitHub | | [Pull Requests](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request) | GitHub documentation on pull requests | |
| **Modification to Project (11/22/23)** | These were the modified steps for NSLDS new joiners:   1. Build a spring boot application, spring boot version 2.2.10 with endpoints that have all CRUD operations. 2. Expand step 1 to add Hibernate layer and connect to H2 in memory database. 3. Build and Angular app using angular 11 4. Connect Angular app to the backend by making http calls to the rest API you build in step 1.   For SPS the most important part is 1 and 2 as we will not have an angular layer for this project, but doesnt hurt to do 3 and 4 as that is standard tech stack across department of ed |

## Task 1 – Develop a new Web Application

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | **Develop a new simple web application** |
| **Conditions** | Given two (2) business weeks, 80 hours,  Use the following tech stack to create an application that meets the following standards |
| **Tech Stack** | * Java 1.8.x ([Download Java](https://www.oracle.com/java/technologies/downloads/#java8)) * IntelliJ IDEA Community Edition 2022.3.3 ([Download IntelliJ](https://www.jetbrains.com/idea/download/other.html)) * Maven 3.x ([Download Apache Maven](https://maven.apache.org/download.cgi?Preferred=ftp://ftp.osuosl.org/pub/apache/)) * Spring Web MVC 4.x or 5.x * JSP * AngularJS 1.6.x (rec. 1.6.9) * Jetty (ex: Jetty Maven Plugin) |
| **Requirements** | 1. ~~Home/Landing page~~ 2. ~~Second page with navigation~~ 3. ~~Two Forms (ex. HTML/Spring <form>)~~    1. ~~Required field~~    2. ~~Optional field~~ 4. ~~Synchronous call (Java MVC)~~ 5. Asynchronous call (AngularJS Service to Rest-style Java Controller) |
| **Optional** | Provide validation for user input in AngularJS AND Jav  ~~Implement better styling with~~ a ~~CSS/Bootstrap~~ |
| **Acceptance Criteria** | Verbal sign-off by team lead after successful demo and code review |
| **Notes** | * Avoid using primitive data types (int, long, boolean), use their wrapper classes instead (Integer, Long, Boolean). * Float and Double can cause weird rounding (example: 1+1=1.999999999) use BigDecimal instead * Unless you are creating custom constructors, you don’t need to define the default constructors in your POJO * Follow the John papa style guide <https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md> * Do NOT copy the example from the Helpful Links. You can use it as a guide but build your own idea. |
| **­Helpful Links** | |  |  | | --- | --- | | [MyLearning – Spring Basics](https://accenture.percipio.com/courses/fc5f25a9-0a68-11e7-89ad-0242c0a80b08/videos/71f9be40-6d85-11e7-9ee2-b6a035da7826) | Maven, Spring Beans, Spring MVC | | [Spring MVC AngularJS example](http://websystique.com/springmvc/spring-mvc-4-angularjs-example/) | Annotations, Restful API, AngularJS, Form | | [AngularJS Style Guide](https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md) | The Team uses this style for readable and maintainable code. | | [Accessibility Fundamentals](https://www.youtube.com/watch?v=z8xUCzToff8) | HTML 508 compliance fundamentals | | [The W3C Markup Validation Service](https://validator.w3.org/#validate_by_input) | HTML markup validation | |

### 

## Task 2 – Persistence Update to Web Application

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | **Augment Web Application (webapp) from Task 1 to integrate with a Database (db)** |
| **Conditions** | Given eight (8) business days, 64 hours  Use the following tech stack to amend your application to meet the following standards |
| **Tech Stack** | * Java 1.8.x * IntelliJ * Maven 3.x * Spring Web MVC 4.2.x or 5.x * JSP * AngularJS 1.6.x (rec. 1.6.9) * Jetty * Hibernate (rec. Hibernate 4, rec. H2 SQL) |
| **Requirements** | 1. ~~Create, Read, Update, Delete via user interaction from the web pages~~ 2. ~~Demonstrate hibernate configuration via @Annotation~~ 3. ~~Execute at least one (1) HQL Query~~ 4. ~~Execute at least one (1) Criteria Query~~ |
| **Optional** | 1. Manager, Service, and DAO Layer have interfaces and implementing classes 2. Create a date input field, this field needs to have JSON convert the input into a data object and submit to the database 3. Display a date on your front end, that comes from your database as a date object |
| **­­Acceptance Criteria** | Verbal sign off by team lead after successful demo and code review |
| **Notes** | * Java controller Layer -> Java Manager Layer -> Java Service Layer -> Java DAO Layer * Dao methods should not return void. * Hibernate configuration must be \*.java * Spring 5.x is not compatible with Hibernate 4.x and below |
| **­Helpful Links** | |  |  | | --- | --- | | [Spring MVC CRUD Example with Hibernate + MySQL](https://www.javaguides.net/2019/12/spring-mvc-crud-example-with-hibernate-jsp-mysql-maven-eclipse.html)  [Install H2 Database and Use H2 Console](https://o7planning.org/11895/install-h2-database-and-use-h2-console)  [H2 database installation on Windows](https://www.learn-it-with-examples.com/database/nosql-databases/h2-database/h2-database-installation-windows.html)  [H2 console view with Spring (not boot)](https://www.javatips.net/blog/how-to-view-content-of-h2-in-memory-database-file-database)  [H2 Database Tutorial and expert Tips](http://www.mastertheboss.com/jbossas/jboss-datasource/h2-database-tutorial)  [Integrate H2 In-memory Database with Spring](https://roytuts.com/integrate-h2-in-memory-database-with-spring/) | Example of Spring MVC CRUD with Hibernate and MySQL (replace mySQL with H2)  How to download H2 console on Windows  How to view content of h2 in-memory or file database with Spring MVC  Tutorial and tips on H2 database  Example code for H2 DB with Spring | | [JPA Criteria Queries](https://www.baeldung.com/hibernate-criteria-queries)  [HQL Queries](https://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html) |  | | [Hibernate Many to Many](https://thoughts-on-java.org/hibernate-tips-map-bidirectional-many-many-association/) | Hibernate Relationships | | [Inheritance Strategies](https://thoughts-on-java.org/complete-guide-inheritance-strategies-jpa-hibernate/) | Useful if there are inheritance relationships in your @Entities | |

## Task 3 – Implement Web Services (REST) for Web Application

|  |  |
| --- | --- |
|  | Description |
| **Task** | **Split Web Application (webapp) from Task 2 into two separate applications. The first application will retrieve information from a REST service provider. The second app will be the service provider built with Spring Boot.** |
| **Conditions** | Given one (1) business week, 40 hours  Use the following tech stack to amend your application to meet the following standards |
| **Tech Stack** | * Java 1.8.x * Eclipse IDE * Maven 3.x * Spring Web MVC 4.2.x or 5.x * Spring Boot 2.x : <https://start.spring.io> |
| **Requirements** | 1. Send and receive data routed through a rest provider (ie. RestTemplate.java)    1. Demonstrate through front end app    2. Demonstrate through Postman touching REST service provider directly 2. Backend app must be built with Spring Boot |
| **Additional Requirements (Optional)** |  |
| **­­Acceptance Criteria** | Verbal sign off by team lead after successful demo and code review |
| **Notes** | * WebApp is split into two full java projects on separate servers, one containing a persistence layer built with Spring Boot and the other containing the frontend UI. |
| **­Helpful Links** | |  |  | | --- | --- | | [Rest Template](https://www.baeldung.com/spring-rest-template-list)  [Spring Boot + H2 CRUD](https://www.bezkoder.com/spring-boot-jpa-h2-example/) | Parameterized list helped for Lists from REST  Step by Step example of REST API with Spring Boot and H2 DB | | [CRUD RestTemplate](http://websystique.com/springmvc/spring-mvc-4-restful-web-services-crud-example-resttemplate/) | RestTemplate -> Restful Controllers | | [Java HTTP Calls](https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/) | More Http Call Examples | | [Many-to-Many Relationship](https://self-learning-java-tutorial.blogspot.com/2020/02/spring-jpa-many-to-many-update-entity.html) [Many-to-Many Relationship - Spring Boot Example](https://asbnotebook.com/jpa-many-to-many-example-spring-boot/) | Spring JPA Many to Many relationship examples | |

## Task 4 – Unit Testing and Web Automation Testing

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | **Add unit testing to Web Application (webapp) and to the REST service application from Task 3. Implement Selenium testing.** |
| **Conditions** | Given two (2) business weeks, 80 hours  Use the following tech stack to amend your application to meet the following standards. |
| **Tech Stack** | * TestNG 6.x or JUnit * Bean Mocking framework, such as Mockito * Selenium |
| **Requirements** | 1. Demonstrate successful execution of unit tests, **both positive and negative testing** 2. Webapp Tests will    1. Open a browser and navigate to your app    2. Fill out and submit a form    3. Verify content of a page 3. Service Layer tests    1. Mock the DAO layer    2. Ensure the service layer performs as expected with 100% code coverage of the Service Layer) 4. For submissions or posts there needs to be validation and error handling on the web app side (see Task 1 required/optional) |
| **­­Acceptance Criteria** | Verbal sign off by team lead after successful demo and code review.  Service layer should have near 100% code coverage. |
| **Notes** | * Webapp tests = Selenium * REST service tests = TestNG * Place all test files in a package on the backed app from Task 3. Task 5 will bypass the frontend app. |
| **`Helpful Links** | |  |  | | --- | --- | | [Installing TestNG](https://testng.org/doc/download.html)  [Selenium + TestNG](https://docs.mendix.com/howto/testing/create-automated-tests-with-testng)  [Add Selenium to POM](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/3.141.59) | Repo for TestNG  Download the drivers individually  Alternate way to add selenium | | [Waits in Selenium Video](https://www.youtube.com/watch?v=K2BRKgZgYPs&list=PLhW3qG5bs-L_8bwNnMHdJ1Wq5M0sUmpSH&index=3)  [Waits in Tests](https://www.selenium.dev/documentation/webdriver/waits/) | Use .waitUntil…. Do not use Thread.sleep() | | [Mocking with Mockito](https://javacodehouse.com/blog/mockito-tutorial/)  [Mockito, The “Spring” way](https://www.baeldung.com/injecting-mocks-in-spring) | Comprehensive example  Mockito | |

## Task 5 – Transition Web Application to Angular

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | **Convert (or duplicate) Web Application (webapp) from task 3 from AngularJS 1.6.x to Angular 8.x** |
| **Conditions** | Given two (2) business weeks, 80 hours  Use the following tech stack to amend your application to meet the following standards |
| **Tech Stack** | * Eclipse IDE * VS Code (Download in Software Center) * Node 12 * Angular 8.x |
| **Requirements** | 1. Webapp will demonstrate all functionality of previous phases |
| **Additional Requirements (Optional)** |  |
| **­­Acceptance Criteria** | Verbal sign off by team lead after successful demo and code review |
| **Notes** | * Rebuilt UI should be functionally identical to previous version (Selenium scripts should still run and pass) |
| **­Helpful Links** | |  |  | | --- | --- | | [Install Node.js and NPM](https://phoenixnap.com/kb/install-node-js-npm-on-windows)  Getting Angular 8 Set Up | Install Node and npm for Windows  You can run **npm install -g @angular/cli@8.3.29** | | [Angular Project Scaffolding](https://itnext.io/choosing-a-highly-scalable-folder-structure-in-angular-d987de65ec7)  [Angular CRUD with Spring Boot](https://www.javaguides.net/2021/08/angular-crud-example-with-spring-boot.html)  [Lazy Loading in Angular 8](https://alekya3.medium.com/how-to-implement-lazy-loading-in-angular-da7761577d23) | Core Feature Shared Modules + examples  Step by step example  Routing and Module Lazy Loading tutorial | | [Core Module vs Shared Module](https://blog.chai-jay.com/angular-core-vs-shared-modules/)  [Sharing between components](https://angularfirebase.com/lessons/sharing-data-between-angular-components-four-methods/) | Deciding where to put components/modules  Sharing data through a Service | | [Angular with Mosh](https://www.youtube.com/watch?v=k5E2AVpwsko)  [Angular with Brad](https://www.youtube.com/watch?v=Fdf5aTYRW0E) | YouTube videos that introduce you to Angular | |

## Task 6 – Additional Requirements (Optional)

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | **Start working in the following tasks. Or reach out to your team lead to see if they have any recommended tasks.** |
| **Sub-Task A** | **Implement your ideas from Task 0** |
|  | Add additional functionality to your app. Everything you add should be held to the same Code standards as previous tasks. |
| **Sub-Task B** | **Make your application to 508 compliant** |
|  | Ensure 508 compliances to make sure that your app is usable by people who are deaf/blind/etc. |
| **Helpful Links** | [508 Checklist](https://www.hhs.gov/web/section-508/making-files-accessible/checklist/html/index.html) |
| **Sub-Task C** | **Implement translation with ngx-translate** |
|  |  |
| **Helpful Links** | [ngx-translate tutorial](https://medium.com/letsboot/translate-angular-4-apps-with-ngx-translate-83302fb6c10d) |
| **Sub-Task D** | **Update New Joiner document** |
|  | Provide your insights for the next group of New Joiners |

# Post-Clearance Phase - Client Web Applications

## Task 1 – Local Workspace for Client Web Applications

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Run StudentLoans.gov (dl-webapp), COD Web (cod-webapp), COD Webservice Provider (cod-ws-prv), COD REST Provider (cod-rest-prv), and eCB (ecb-webapp) in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Oracle / Linux account with access to ORACLE TDS instance, ACDOE account with access to Subversion repo cod and Confluence  Given one (1) business week, 40 hours |
| **Standards** | Must demonstrate the successful execution of StudentLoans.gov, COD Web, and eCB in a local workspace to include:  Sites are accessible  Sites are accessible for authentication |

## Task 2 Access – Simulated Change Request for Client Web Applications

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Perform mock client change request upgrades for StudentLoans.gov (dl-webapp), COD Web (cod-webapp), COD Webservice Provider (cod-ws-prv), COD REST Provider (cod-rest-prv), and/or eCB (ecb-webapp) in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Oracle / Linux account with access to ORACLE TDS instance, ACDOE account with access to Subversion repo cod and Confluence  Given mock requirement(s) *Mock requirements will be determined at phase beginning*  Given one-and-a-half (1.5) business week, 60 hours |
| **Standards** | Must demonstrate the successful implementation of the mock requirement(s) for StudentLoans.gov, COD Web, and/or eCB in a local workspace  An approved code review is required for this successful completion of this phase |

## Task 3 Access – Repair Client Web Applications

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Perform troubleshooting for StudentLoans.gov (dl-webapp), COD Web (cod-webapp), COD Webservice Provider (cod-ws-prv), COD REST Provider (cod-rest-prv), and/or eCB (ecb-webapp) in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Oracle / Linux account with access to ORACLE TDS instance, ACDOE account with access to Subversion repo cod and Confluence  Given a defective version of the web application(s) *Defects will be introduced the local workspace at phase beginning* Local workspace will be disconnected from subversion after defect(s) introduction  Given one-and-a-half (1) business week, 60 hours |
| **Standards** | Must demonstrate the successful resolution of the defect(s) for StudentLoans.gov, COD Web, and/or eCB in a local workspace  An approved code review is required for this successful completion of this phase |

# Post-Clearance Phase – TIVOD Testing Tools

## Task 1 – Local Workspace for Testing Tools

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Run Automation JAR (auto-unit), Automation webapp (auto-web & auto-api & auto-auth-server), and MITE (mite-ui) in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Oracle / Linux account with access to ORACLE TDS instance, ACDOE account with access to Subversion repos **rft** and **techarch**, and Confluence  Given one (1) business week, 40 hours |
| **Standards** | Must demonstrate the successful execution of Automation JAR, Automation webapp, and MITE in a local workspace to include:  Sites are accessible  Sites are accessible for authentication  Successful script execution |

## Task 2 Access – Simulated Change Request for TIVOD Testing Tools

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Perform mock change request upgrades for Automation JAR (auto-unit), Automation webapp (auto-web & auto-api & auto-auth-server), and/or MITE (mite-ui) in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Oracle / Linux account with access to ORACLE TDS instance, ACDOE account with access to Subversion repos **rft** and **techarch**, and Confluence  Given mock requirement(s) *Mock requirements will be determined at phase beginning*  Given one-and-a-half (1.5) business week, 60 hours |
| **Standards** | Must demonstrate the successful implementation of the mock requirement(s) for Automation JAR, Automation webapp, and/or MITE in a local workspace  An approved code review is required for this successful completion of this phase |

## Task 3 Access – Repair TIVOD Testing Tools

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Perform troubleshooting for Automation JAR (auto-unit), Automation webapp (auto-web & auto-api & auto-auth-server), and/or MITE (mite-ui) in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Oracle / Linux account with access to ORACLE TDS instance, ACDOE account with access to Subversion repos **rft** and **techarch**, and Confluence  Given a defective version of the application(s) *Defects will be introduced the local workspace at phase beginning* Local workspace will be disconnected from subversion after defect(s) introduction  Given one-and-a-half (1) business week, 40 hours |
| **Standards** | Must demonstrate the successful resolution of the defect(s) for Automation JAR, Automation webapp, and/or MITE in a local workspace  An approved code review is required for this successful completion of this phase |

# Post-Clearance Phase – DCC Testing Tools

## Task 1 – Learn SeeTest

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Become familiar with SeeTest and its tools |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), and access to Trial SeeTest server  Given two (2) business days, 24 hours |
| **Standards** | Must demonstrate familiarity with the cloud site   * Log into dashboard * Check out device and utilize Object Spy   Must demonstrate ability to connect to SeeTest via Appium Studio in Eclipse   * Run sample automation code and view output in cloud site * Check out device and utilize Object Spy |
| **Helpful Links** | <https://docs.experitest.com/display/TE/Test+Execution+Home>  <https://docs.experitest.com/display/TC/Test+Development+Home> |

## Task 2 – Local Workspace for DCC Testing Tools

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Run Automation project in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Appium Studio and TestNG plugins for Eclipse, ACDOE account with access to GIT repo **dcc-tests** and the DCC SeeTest server, and Teams  Given two (2) business days, 16 hours |
| **Standards** | Must demonstrate the successful execution of the Automation project locally |

## Task 3 – Sample Flow Creation

|  |  |
| --- | --- |
|  | **Description** |
| **Task** | Create sample page and tests objects for the Automation project in a local workspace |
| **Conditions** | Given an AFS laptop with Java 1.8.x, Eclipse IDE (minimum of Neon.3), Maven (embedded with Eclipse or standalone Maven 3.x), Appium Studio and TestNG plugins for Eclipse, ACDOE account with access to GIT repo **dcc-tests** and the DCC SeeTest server, and Teams  Given mock requirement(s) *Mock requirements will be determined at phase beginning*  Given one (1) business week, 40 hours |
| **Standards** | Must demonstrate the successful implementation of the mock requirement(s) for Automation project in a local workspace  An approved code review is required for this successful completion of this phase |