

---

# **CoachEra Maintainer Manual**

---

Rayan Wali, Yann Hicke, Jayesh Paunikar, Elena Stoeva, Ashley Yu,  
Hehong Li, Angela Lau, Yixin Zhang

## I. Requirements Analysis and Specification

The key high-level requirement that the client had for us is to extend or modify defined functionalities of the MobileCoach chatbot system. The client provided us with the following specific functionalities that they were looking for in the final release of the system:

- A dynamic dashboard that includes a star system, a progress indicator that collectively reflect the progress of a user, a weekly task list, and an indicator of past weekly accomplishments.
- A Frequent Questions and Answers page that is focused on guiding users on how the application can help them achieve their goal.
- Conversational flows that define the interaction between the chatbot and the user.
- Push notifications that could be sent from the mobile application to the user's device.

The requirements listed above are the key requirements. There were certainly many sub-requirements within each point above that the client defined and that we resolved. However, there are also requirements that the client tentatively stated but had to let go of due to other priorities. Our expectation is that future maintainers like you can take these functionalities we implemented following the client's specifications and extend them to turn the application into one that is for all types of learners and for different courses, which is something we were not able to implement, but is something we can expect developers to implement on top of our complete package, containing the complete code and documentation, in the future.

## II. Test Facilities

### A. Test Plan

We test the components that we implemented in the MobileCoachApp repository using the following approaches:

- a. Unit tests (including snapshot tests and functionality unit tests)
- b. End-to-end tests
- c. Continuous Integration through Jenkins

### B. Unit Tests

We implement both functionality tests and snapshot tests using Jest - a JavaScript test runner.

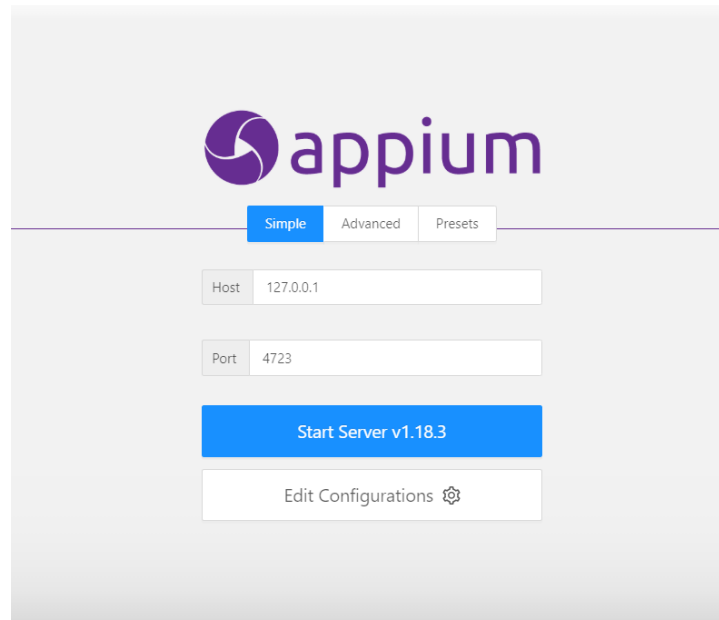
Functionality unit tests are assertion tests for the methods and classes that we implemented and snapshot tests ensure that UI does not change unexpectedly. To run

all tests inside repository, run the command 'yarn test' assuming that you have installed all dependencies with 'yarn install' Note that when you change any component in the application, you need to first run 'yarn test -u' which will update the reference snapshot for the snapshot tests.

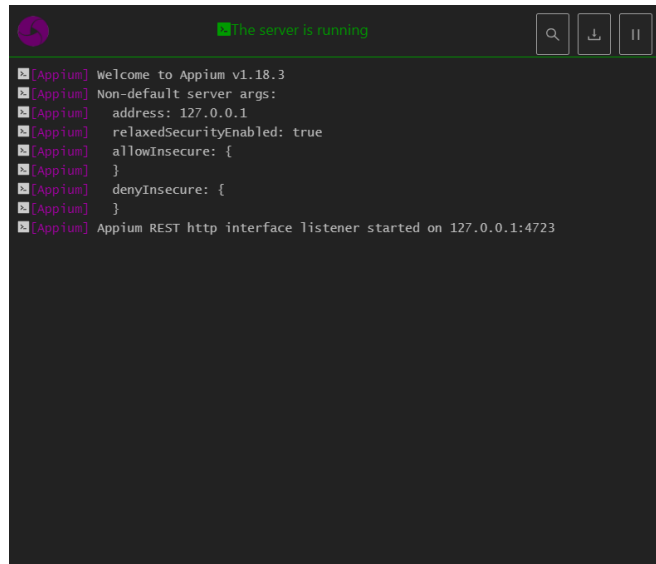
All tests for dashboard in '\_test\_' folder under 'containers' folder. The original mobilecoach project does not have any test, and all unit tests added by us are in that folder.

### C. End-to-End Tests

Appium and Selenium are two automated testing frameworks we use to test our application, we integrated these two with the android emulator. First, we start a local server at the url: <http://127.0.0.1> using Appium and the configurations are set to the path of JAVA and Android in your own laptop.

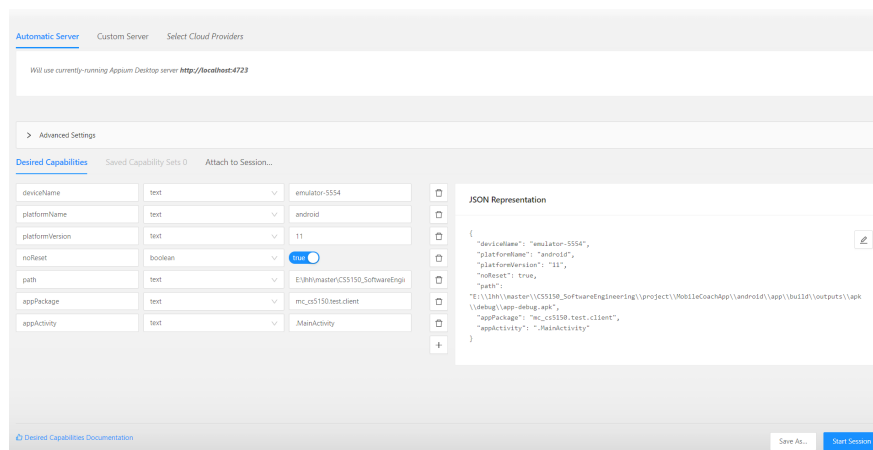


And you could also set the host address to 0.0.0.0, that will also start a local Appium Server, but you need to use different addresses which you use here for the future connection of your java program and Appium.



```
[Appium] welcome to Appium v1.18.3
[Appium] Non-default server args:
[Appium]   address: 127.0.0.1
[Appium]   relaxedSecurityEnabled: true
[Appium]   allowInsecure: {
[Appium]   }
[Appium]   denyInsecure: {
[Appium]   }
[Appium] Appium REST http interface listener started on 127.0.0.1:4723
```

Then we connect the Appium server and our Android simulator by entering the parameter of the application. For example, `deviceName`, `platformName`, `platformVersion`, `appPackage`, `appActivity` and `path` (.apk). To get the values of these, you can run `'adb devices'` for the `deviceName` and `dumpsys window windows | grep -E 'mCurrentFocus'` under the command `'adb shell'` on your cmd if you are using a Windows system.



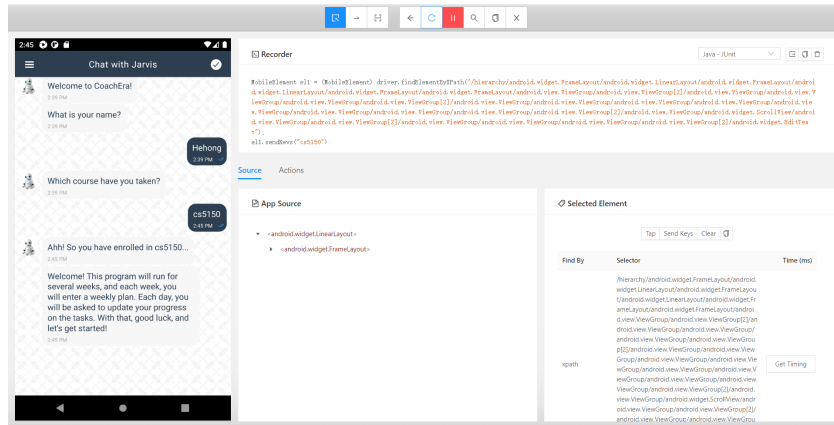
The screenshot shows the 'Desired Capabilities' configuration window in Appium Desktop. It includes a table for setting capabilities and a JSON representation of the configuration.

Capability	Type	Value
deviceName	text	emulator-5554
platformName	text	android
platformVersion	text	11
noReset	boolean	True
path	text	E:\IshMaster\CS5130_SoftwareEngi
appPackage	text	mc_cs5130.test.client
appActivity	text	MainActivity

**JSON Representation**

```
{
  "deviceName": "emulator-5554",
  "platformName": "android",
  "platformVersion": "11",
  "noReset": true,
  "path": "E:\\IshMaster\\CS5130_SoftwareEngineering\\project\\RobotCoachApp\\android\\app\\build\\outputs\\apk\\debug\\app-debug.apk",
  "appPackage": "mc_cs5130.test.client",
  "appActivity": ".MainActivity"
}
```

If it is connected successfully, you can get the following page showing the same screen with our emulator. Then we use the internal function shown at the top of the page to record the actions we would like to test based on the scenarios we planned to perform before. We added some clicking and texts which we want to chat with the e-coach.



And when we get all the functions with the element number and the actions on them and these are all added into the java program. And then you need to download 4 basic libraries containing the classes used for the program, which are *selenium-server-standalone-3.141.59*, *java-client-7.0.0*, *selenium-java-3.141.59*, *commons-lang3-3.12.0*. You may be able to do the latest version, but it may need some configuration changes, for example I was having an error for 'MobileElement' is changed to 'WebElement' in the latest java-client packages.

Besides, in your Jjava program, you also need to set these parameters to make the connection between your program and the emulator through Appium. These are the same values with the ones you use to connect with Appium.

```
DesiredCapabilities dc= new DesiredCapabilities();
dc.setCapability(MobileCapabilityType.DEVICE_NAME, "emulator-5554");
dc.setCapability("platformName", "android");
dc.setCapability("appPackage", "mc_cs5150.test.client");
dc.setCapability("appActivity", ".MainActivity");
dc.setCapability("app", "E:\\lhh\\master\\CS5150_SoftwareEngineering\\project\\AwesomeProject\\android\\app\\build\\outputs\\
//dc.setCapability("ms:waitForAppLaunch", 25);
```

Finally, after following all of these steps, when you run your java program, you will see the application is doing all of the pre-set actions automatically. There is one thing to notice is that if your app reacts slowly or needs some more time to launch at the starting part, please add some sleep time for the thread.

### III. Continuous Integration

We maintain continuous integration through Jenkins which is installed and runs on our server. Every push and pull request to the MobileCoachApp repository triggers a new build in our Jenkins project which runs all unit tests in the repository. If a build fails, an email notification is sent to a selected email address. You can log into our Jenkins environment through this URL: <http://132.236.91.14:8081/>

The Jenkins project that creates builds that run our unit tests is called MobileCoachApp.

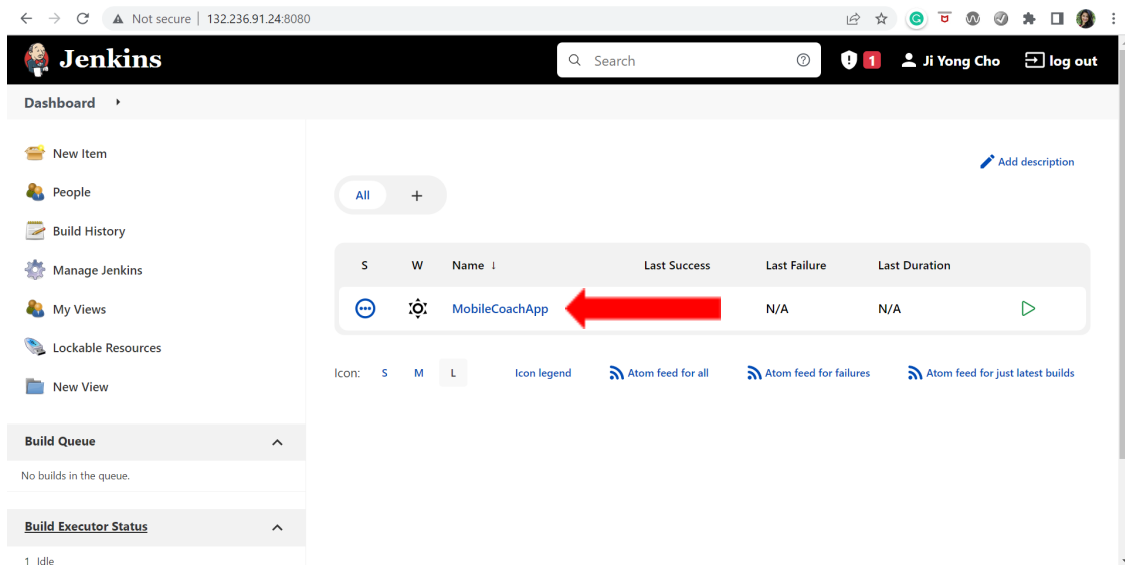
### Getting Started With Jenkins:

Jenkins is installed on the production server and it is accessible at

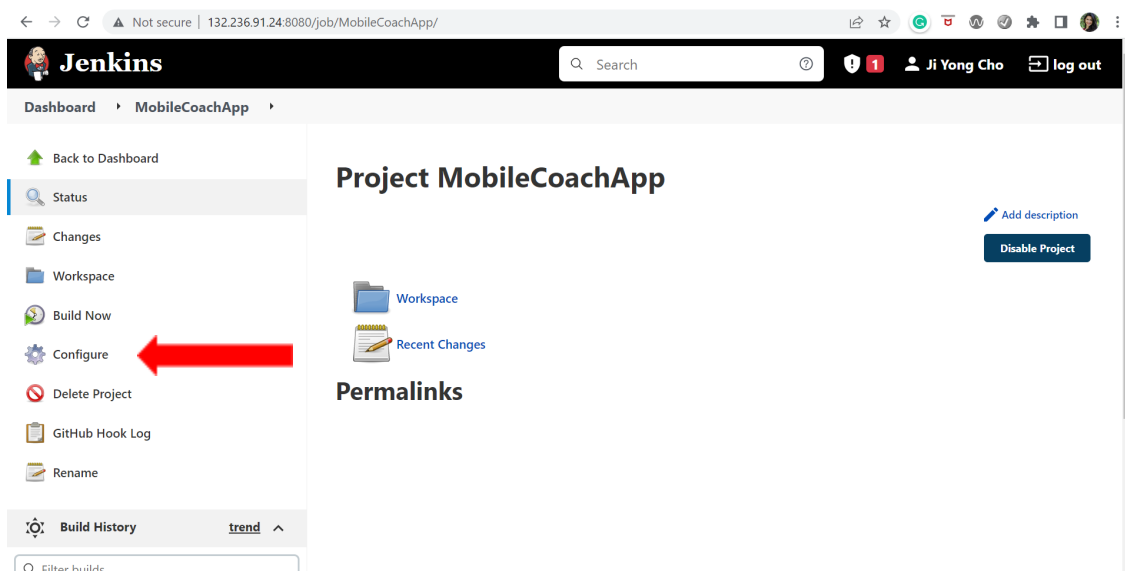
<http://132.236.91.24:8080/>

Please contact Ji Yong Cho for login credentials.

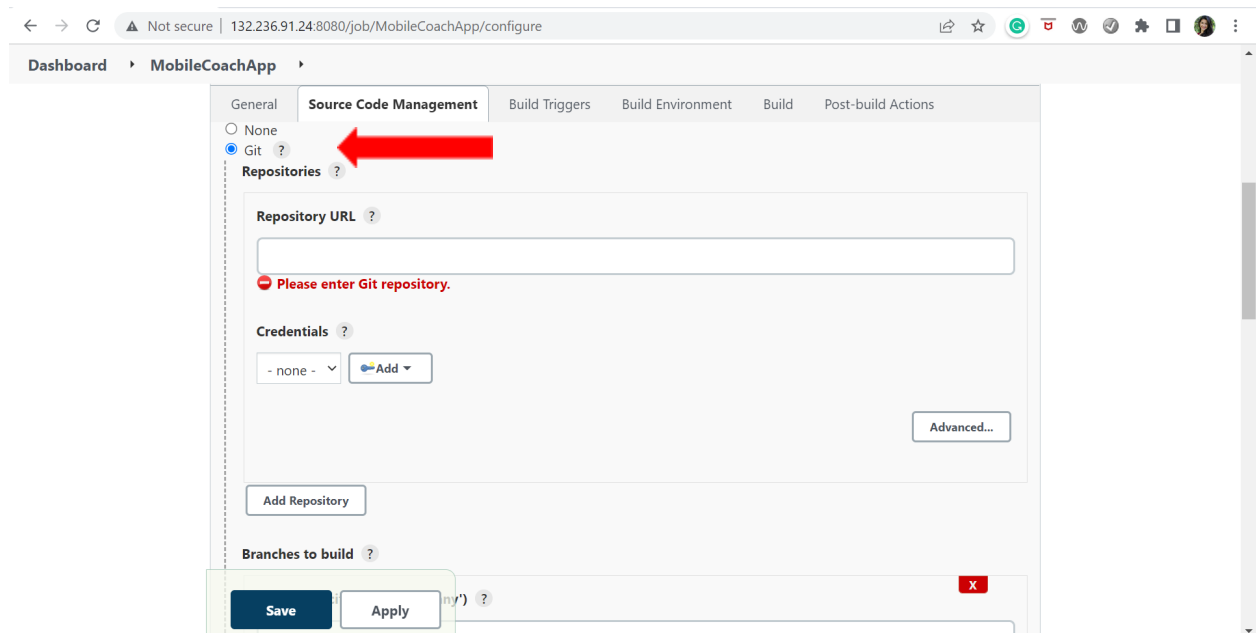
Once you log in, you will see that we have created a Jenkins Project called MobileCoachApp.



Click on the name of this project. Then select the "Configure" button in the menu to the left.

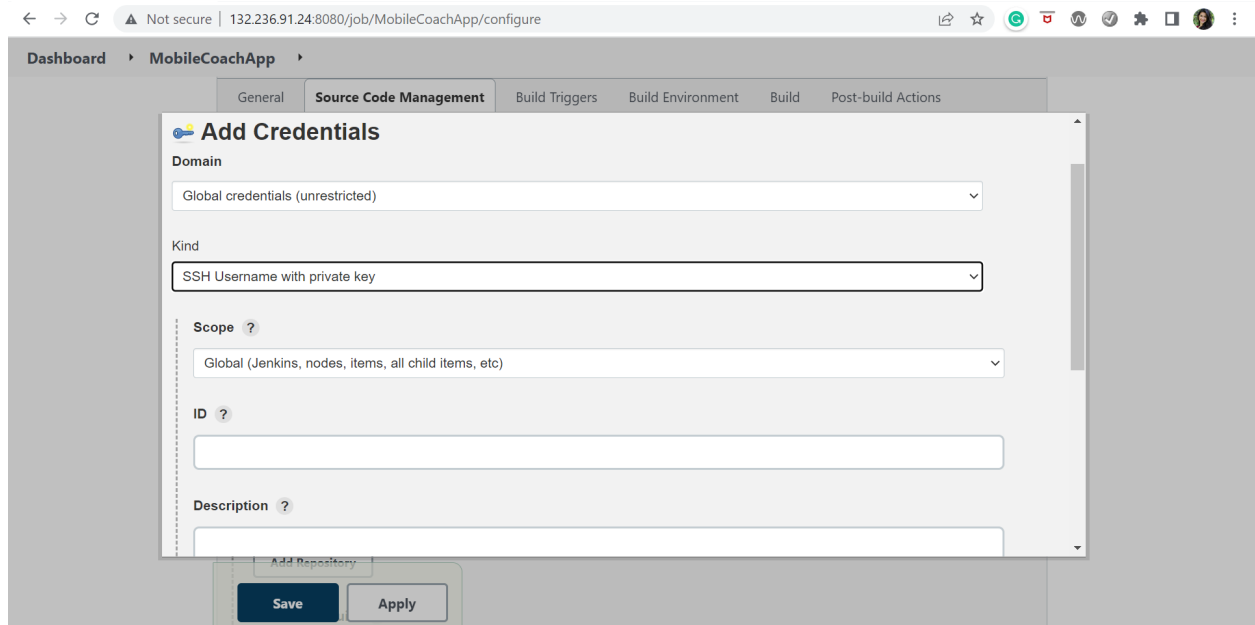


Under the “Source Code Management” tab, select the “Git” option.



The screenshot shows the Jenkins configuration page for a job named 'MobileCoachApp'. The 'Source Code Management' tab is selected. Under the 'Source Code Management' section, the 'Git' option is selected with a radio button. A red arrow points to the 'Git' option. Below this, there is a 'Repositories' section with a 'Repository URL' field and a 'Credentials' dropdown menu. A red error message 'Please enter Git repository.' is displayed below the 'Repository URL' field. The 'Credentials' dropdown is set to '- none -' and has an 'Add' button next to it. There is also an 'Advanced...' button. At the bottom, there is an 'Add Repository' button and a 'Branches to build' section. The 'Save' and 'Apply' buttons are at the bottom right.

If the repository that you are working on is public, just enter its http or ssh address. If it is private. You need to provide Git credentials. Click on the “Add” button under the “Credentials” tab.



The screenshot shows the 'Add Credentials' dialog box in Jenkins. The 'Domain' dropdown is set to 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'SSH Username with private key'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. There is an 'ID' field and a 'Description' field. At the bottom, there are 'Save' and 'Apply' buttons.

We recommend that you select SSH Username with private key under “Kind”.

The screenshot shows the Jenkins configuration page for 'MobileCoachApp' under the 'Source Code Management' tab. A modal window is open for adding a new source code management provider. The fields include 'Description', 'Username', 'Private Key' (with an option to 'Enter directly'), and 'Passphrase'. The 'Add' button is highlighted in blue, and the 'Cancel' button is in grey. Below the modal, the 'Save' button is also highlighted in blue.

Write your GitHub username in the “Username” field. Select “Enter directly” under “Private Key”, click on the “Add” button, and enter your **private** SSH key. Then click on the “Add” button to add your SSH credentials. Note: if you use SSH credentials, you need to add the **SSH URL** of your repository, not the HTTP one.

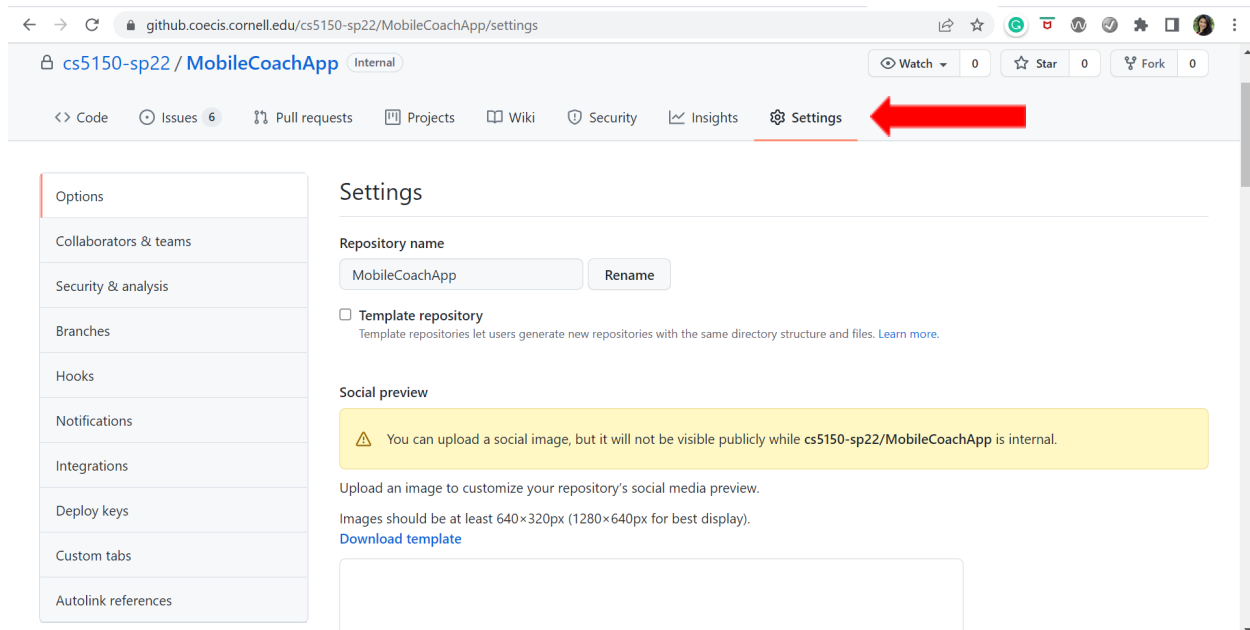
(Optional) Under “Post-build Actions” you can choose “Email Notification” and provide an email address to which an email notification will be sent every time when a build fails (this is the approach that we used in our CS5150 team). If you decide to have this functionality, you also need to perform some additional setup in order to enable Jenkins to send emails to your email address. Please follow the instructions on [this page](#).

The screenshot shows the Jenkins configuration page for 'MobileCoachApp' under the 'Post-build Actions' tab. The 'E-mail Notification' section is expanded, showing the 'Recipients' field (with a description: 'Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.') and the 'Send e-mail for every unstable build' checkbox, which is checked. The 'Add post-build action' button is visible at the bottom. The 'Save' button is highlighted in blue.

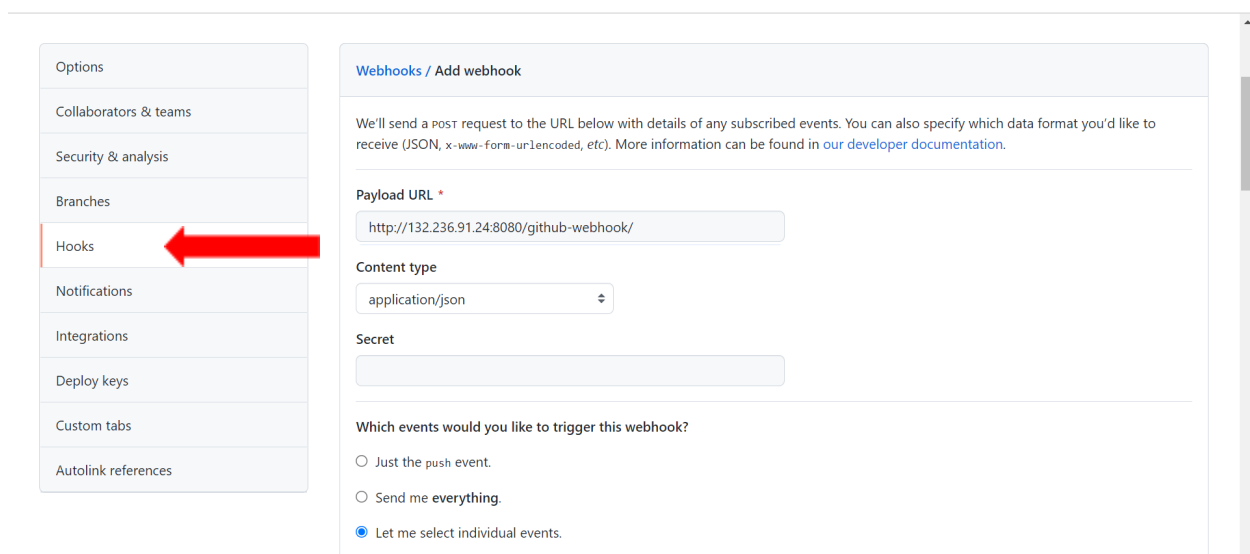


Finally, click on the “Save” button. Your Jenkins project is now configured.

Now, head over to your repository and click on the “Settings” tab.



Click on the “Hooks” tab in the menu to the left. Then click on the “Add webhook” button. Under “Payload URL” type in <http://132.236.91.24:8080/github-webhook/>. Choose “application/json” under “Content type”. Under the question “Which events you like to trigger this webhook?”, select “Let me select individual events.” From the events, select “Pushes” and “Pull requests”.



<input type="checkbox"/> <b>Projects</b> Project created, updated, or deleted.	<input type="checkbox"/> <b>Pull request review comments</b> Pull request diff comment created, edited, or deleted.
<input type="checkbox"/> <b>Pull request review threads</b> A pull request review thread was resolved or unresolved.	<input type="checkbox"/> <b>Pull request reviews</b> Pull request review submitted, edited, or dismissed.
<input checked="" type="checkbox"/> <b>Pull requests</b> Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, synchronized, ready for review, converted to draft, locked, unlocked, auto merge enabled, auto merge disabled, milestone, or demilestoned.	<input checked="" type="checkbox"/> <b>Pushes</b> Git push to a repository.
<input type="checkbox"/> <b>Registry packages</b> Registry package published or updated in a repository.	<input type="checkbox"/> <b>Releases</b> Release created, edited, published, unpublished, or deleted.
<input type="checkbox"/> <b>Repositories</b> Repository created, deleted, archived, unarchived, publicized, privatized, edited, renamed, or transferred.	<input type="checkbox"/> <b>Secret scanning alerts</b> Secrets scanning alert created, resolved, or reopened.
<input type="checkbox"/> <b>Stars</b> A star is created or deleted from a repository.	<input type="checkbox"/> <b>Statuses</b> Commit status updated from the API.
<input type="checkbox"/> <b>Team adds</b> Team added or modified on a repository.	<input type="checkbox"/> <b>Visibility changes</b> Repository changes from private to public.

Finally, click on the “Add Webhook” button.

Now, your Jenkins CI should be set up. You can test it by making a small change in the repository (e.g. changing the README file) and pushing it. Then go to the Jenkins project that we configured above and look at the “Build History” section on the left. There should be a new build. You can click on the build to see more details. For example, after you click on the build, you can click on the “Console” button to see all commands that were executed and their logs.

Dashboard
MobileCoachApp
Back to Dashboard

Status
Changes
Workspace
Build Now
Configure
Delete Project
GitHub Hook Log
Rename
Build History
trend
Filter builds...
#45 May 9, 2022, 12:44 AM

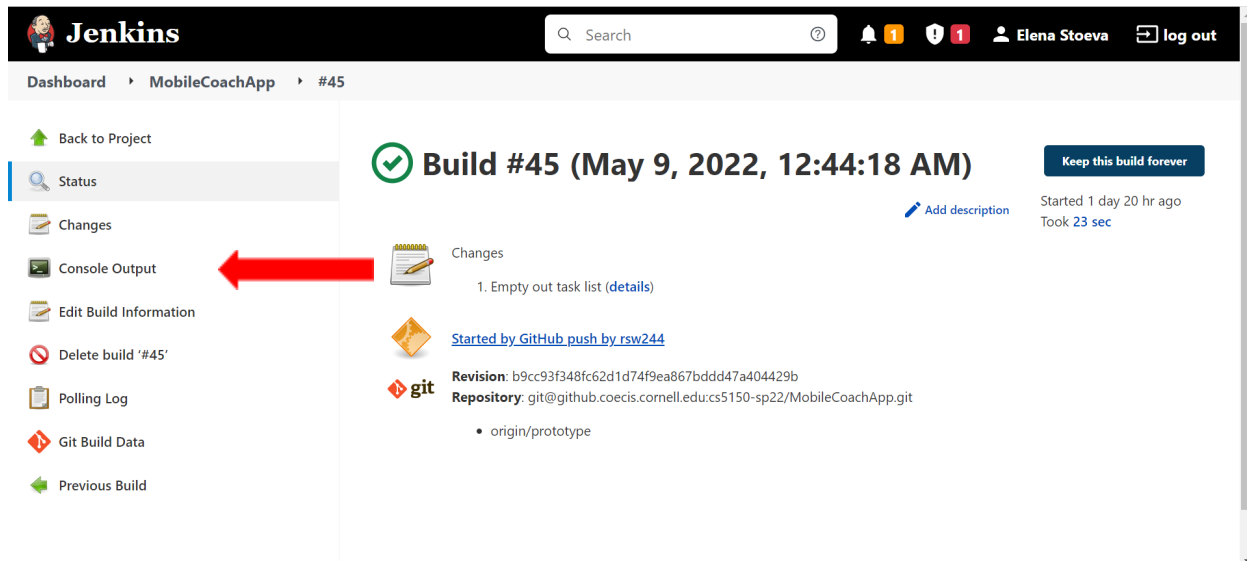
## Project MobileCoachApp

Add description
Disable Project

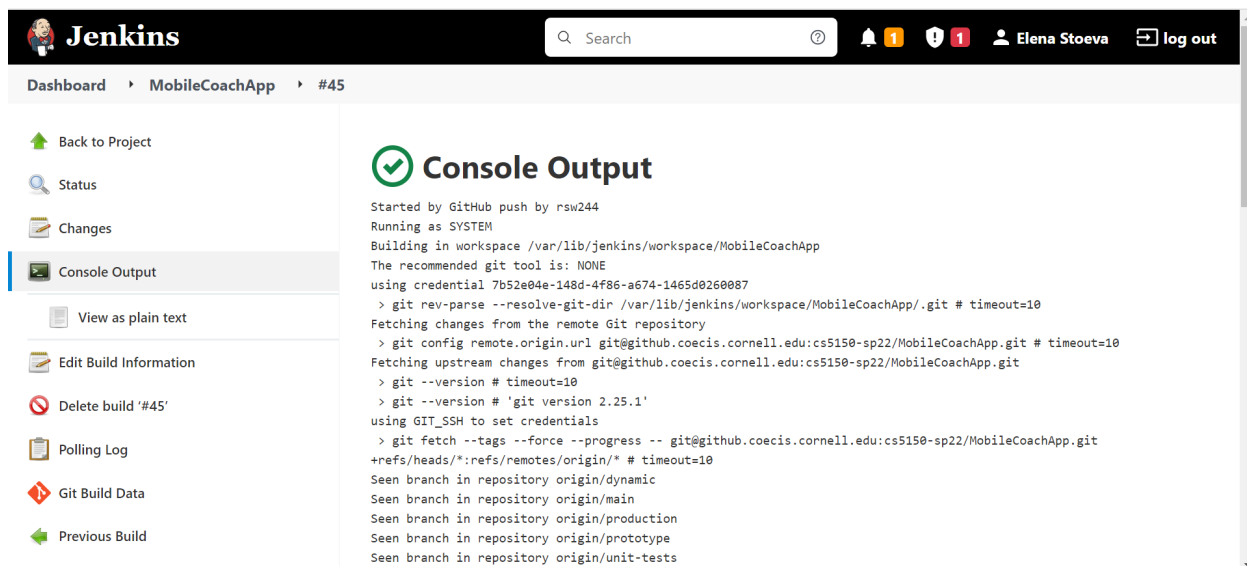
Workspace
Recent Changes

### Permalinks

- Last build (#45), 1 day 20 hr ago
- Last stable build (#45), 1 day 20 hr ago
- Last successful build (#45), 1 day 20 hr ago
- Last failed build (#43), 2 days 0 hr ago
- Last unsuccessful build (#43), 2 days 0 hr ago
- Last completed build (#45), 1 day 20 hr ago



The screenshot shows the Jenkins interface for Build #45 of the MobileCoachApp project. The left sidebar contains a list of links: Back to Project, Status, Changes, Console Output, Edit Build Information, Delete build '#45', Polling Log, Git Build Data, and Previous Build. A red arrow points to the 'Console Output' link. The main content area displays the build status as 'Build #45 (May 9, 2022, 12:44:18 AM)' with a green checkmark icon. It includes a 'Keep this build forever' button, a description link, and build statistics: 'Started 1 day 20 hr ago' and 'Took 23 sec'. Below this, it shows the build was 'Started by GitHub push by rsw244' and provides the revision and repository information.



The screenshot shows the Jenkins interface for the Console Output of Build #45. The left sidebar is the same as the previous screenshot, but the 'Console Output' link is highlighted. The main content area displays the console output, which includes the following text:

```

Started by GitHub push by rsw244
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/MobileCoachApp
The recommended git tool is: NONE
using credential 7b52e04e-148d-4f86-a674-1465d0260087
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/MobileCoachApp/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@github.coecis.cornell.edu:cs5150-sp22/MobileCoachApp.git # timeout=10
Fetching upstream changes from git@github.coecis.cornell.edu:cs5150-sp22/MobileCoachApp.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_SSH to set credentials
> git fetch --tags --force --progress -- git@github.coecis.cornell.edu:cs5150-sp22/MobileCoachApp.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/dynamic
Seen branch in repository origin/main
Seen branch in repository origin/production
Seen branch in repository origin/prototype
Seen branch in repository origin/unit-tests

```

#### IV. Revised System Design

For our project, we adopted an iterative refinement methodology, where we would iterate upon the features that we developed. Combined with a greater knowledge of our system components and interactions, we would like to produce a revised version of our system design, which we hope is helpful for maintainers of this system for getting a grasp of the system in a simpler manner than what was provided to us. In Figure 1 (UML Deployment Diagram), we identify the different components of the architecture of the system we deployed: apps on phones, Web app on a computer, four docker images on a Linux server (tomcat, deepstream, nginx and mongodb). All these services communicate with one another

over HTTPS to both render an app on a phone and a controlling web app for interventions on a laptop.

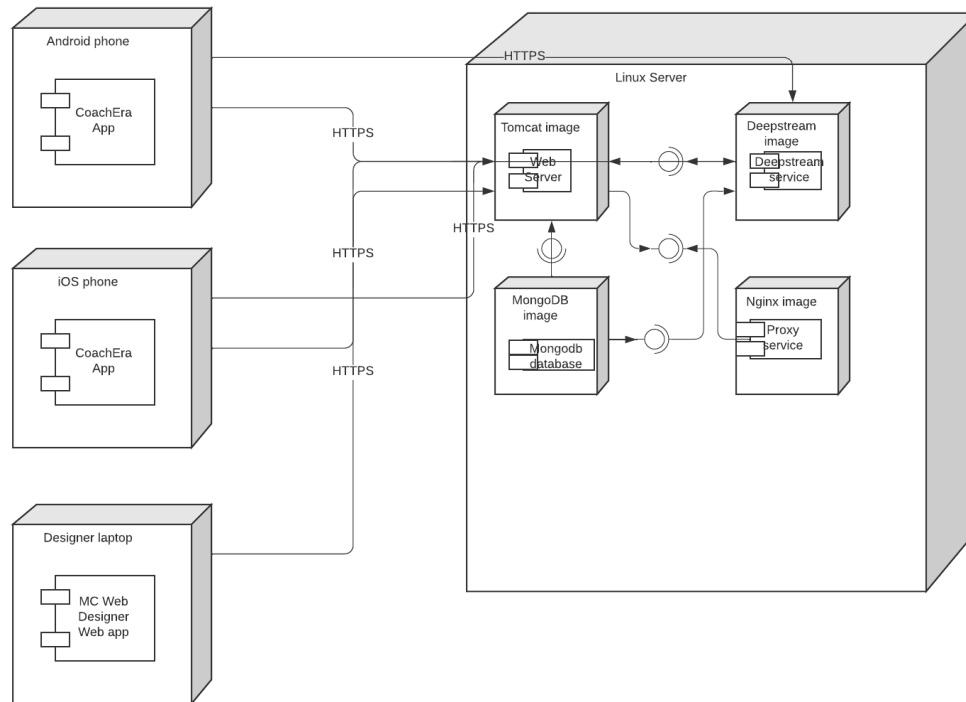


Figure 1: UML Deployment Diagram

**Note:** the capacity of the MongoDB is limited. After 500 new participants are added the server will start to slow down. Some support from Cornell IT is required.

## V. Style Guide

A *style guide* is a documentation of standards that should be followed if chosen to be applied to a codebase. The MobileCoach codebase, particularly the code for the newly implemented features (Dashboard, Chat, FAQ), follows the Google JavaScript style guide, which can be found at <https://google.github.io/styleguide/jsguide.html>. This style guide contains different conventions that programmers like you should follow if they plan on writing additional JavaScript code to extend existing features made by our team.

For example, Section (3.4.1) of the style guide corresponds to following a certain convention when writing import statements. The style guide specifies that a `.js` extension *must* be added to the end of the file path in order to be compliant with the style guide. It is thus good practice to review the style guide before you begin writing code so that you do not have to make significant style modifications when you are done writing code, and particularly when you anticipate on writing lots of lines of code.

Please note that the conventions in this style guide are only applied to the modifications that were made to the initial state of the MobileCoach repository. If you plan on applying the rules defined in the style guide across all components of the codebase, please ensure that all the code that you plan on modifying is consistent with the other parts of the codebase.

## VI. Future UI Designs

We do foresee potential changes that could be made to improve the user interface of our system. For example, there are improvements we thought about for the Dashboard page, including layout of different components on the screen, accessibility, and usefulness.

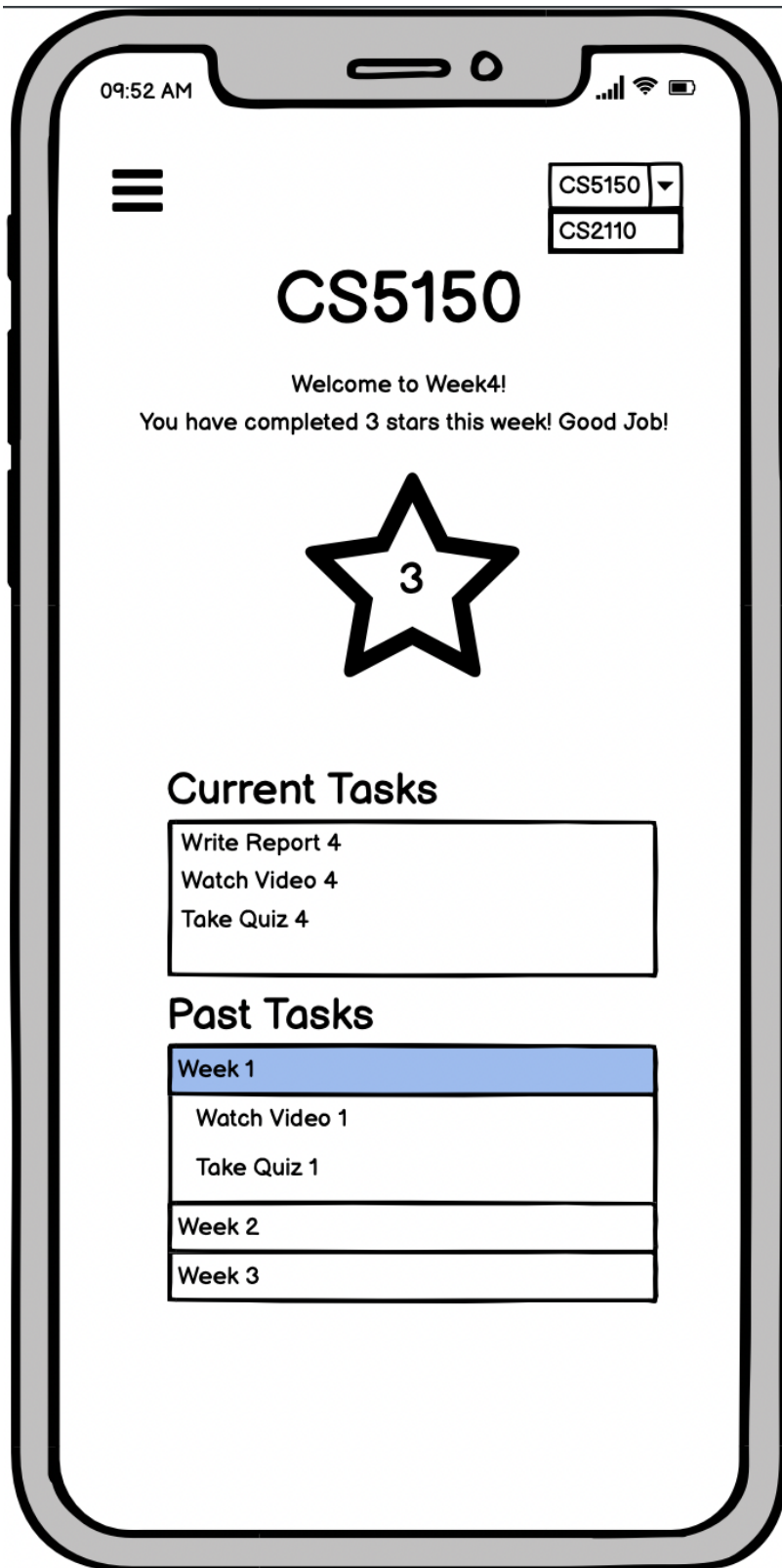
In this section, we propose some of the improvements that we had in mind regarding the user interface of the application.

### A. Chat

The chat screen is mostly provided by the original MobileCoach app. Current screen conforms to the convention for chatbot interface and is functioning as we intended. We do not propose a new interface for this screen to fit our users' existing mental model as much as possible. If there are new features added in the future, corresponding changes can be made to this screen to accommodate the new functionalities.

### B. Dashboard

The dashboard screen is primarily designed and implemented by our team. It has been through several iterations. We propose a new iteration for the layout according to our latest user feedback, as shown below.



09:52 AM



CS5150

CS2110

# CS5150

Welcome to Week4!

You have completed 3 stars this week! Good Job!



## Current Tasks

Write Report 4

Watch Video 4

Take Quiz 4

## Past Tasks

Week 1

Watch Video 1

Take Quiz 1

Week 2

Week 3

In this new design, we added the course choosing screen so it supports tracking of multiple courses at the same time. The stars are tracked using a big star icon instead of a line of stars to provide a clearer representation of achievements, which also gamify the star earning process. Current tasks and past tasks are wrapped in blocks for a better appearance. Also for past tasks, users can now check specifically what they have accomplished with the accordion instead of just the number of stars.