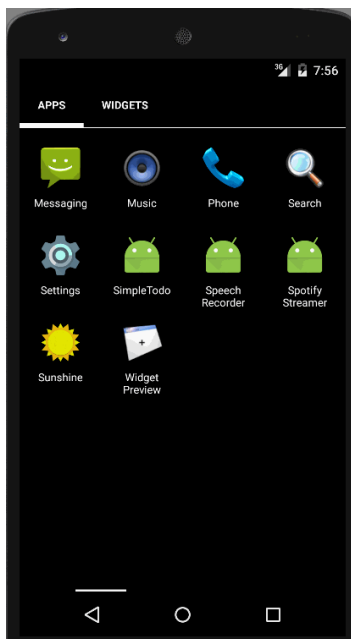


Android Pre-work: Todo App

The pre-work incorporates a few steps:

1. Setup your Android environment
2. Build initial todo app
3. Submit todo app for review via Github
4. Add edit functionality to todo app
5. Extend your todo app, improve UI, add features



Today's assignment will take you through a complete local setup and building your first app, a simple todo list. The guide material for this assignment is divided into two main sections:

1. **Setup** of your local workstation with Android Studio and all necessary tools
2. **SimpleTodo** app design, development, and submission via Github

These two sections are further broken down into several chapters, each of which is covered in a video with accompanying slides. Proceed through all chapters in the order presented, making sure you understand the concepts and have completed the objectives presented in each chapter before moving on to the next.

Running into issues with your app, such as crashes? Check out our solving problems in Android apps guide to learn about how to debug and correct issues as they come up. You can also reference this list of common issues specific to this assignment.

1. Setup Android

In the first section, you will install and configure your local workstation for Android development, including all necessary tools such as **Android Studio** and **Git**, including configuring the emulator, setting up your first Android project, and using Git. You can follow the whole setup playlist here. Completing the setup can often take ~90 minutes. The steps for setup are outlined in more detail below:

1.1 - Install Android Studio

- **MacOS**
 - 🎬 Install Android Studio (7:42)
 - Slides
 - Download and install Android Studio.
- **Windows**
 - 🎬 Windows - Android Studio Setup (8:22)
 - Slides
 - Download and install Android Studio.

1.2 - Git & First Android Project

- 🎬 Git & First Android Project (7:58)
 - Slides

Introduction to Git and GitHub and basic Git commands. Cloning a sample project from GitHub and opening and running it in Android Studio.

1.3 Emulator

- 🎬 Emulator (7:07)
 - Slides
- Running Apps on Your Device

Set up the Android emulator and learn how to run apps on it. Learn about running and testing apps on a physical device. Change the default layout template in Android Studio.

2. Build the SimpleTodo App

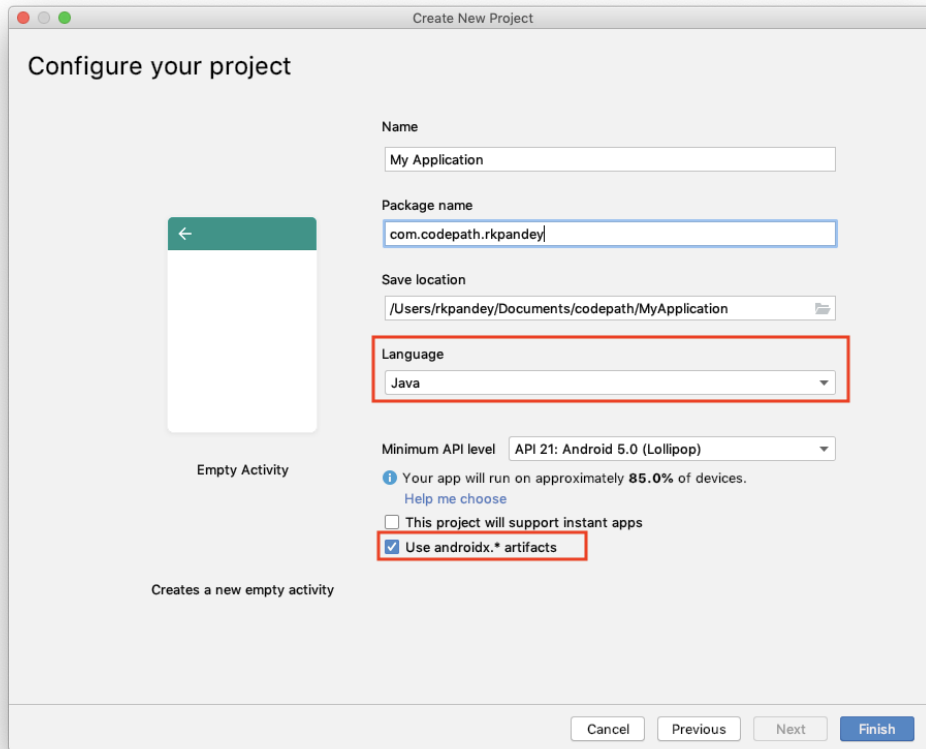
In this section, you will design, develop, and submit your first app: a simple todo list manager. You'll understand how to turn requirements into a design, wireframe a UI, and devise a strategy for systematically implementing features step-by-step. At the end, you'll learn how to publish your code to Github and submit your project via the CodePath course portal. The steps you learn in these videos will be used throughout the rest of the course.

Completing and then submitting the initial todo app will likely take a few hours. You can build the app by coding along with this video playlist. The steps are outlined in more detail below:

2.1 - Setup

- 🎬 SimpleTodo Setup (6:55)

- Be sure to select the box to use `androidx.*` artifacts.
- Because all submissions will be done in Java, make sure to disable Kotlin support by unchecking the box when creating a new Android Studio project.



- If you project starts up with Kotlin files (i.e. `MainActivity.kt`), you should start over and create a new project with this checkbox unselected.

- Android Directory Structure
- Organizing your Source Files

Begin by creating a new project for the SimpleTodo app, understanding the which minimum API version to use and how projects in Android Studio are organized.

- Objectives
 - New, runnable project set up from scratch
 - Understand minimum API version
 - Understand project layout and key files

2.2 - Design

- 📺 SimpleTodo Design (15:51)
- Constructing View Layouts
- Defining Views and their Attributes

In this step, you'll review the required features for your app and translate them into a strategy for implementation, including how to think about different areas of functionality using the Model-View-Controller. You'll see how to wireframe the app and turn that into a user interface using the Android XML layout designer.

- Objectives

- Understand required features / stories
- MVC pattern
- Wireframe user interface
- Build user interface via layout
- Understand layout Design vs. Text views

2.3 - Render the list of items

- 🎬 SimpleTodo Rendering Items (17:13)
- Understanding App Resources
- Understanding Activity Lifecycle

In this step, we'll write Java code which manages the data model and adapter, and wires the list of todo items in a list using a RecyclerView.

- Objectives
 - Understand basics of Java coding
 - Extending `onCreate()`
 - Adding new methods to activity class
 - Defining a data model
 - Defining an adapter and viewholder
 - Using Android layouts
 - Binding the adapter to the data source

2.4 - Implement add and remove functionality

- 🎬 SimpleTodo Adding/Removing Items (17:35)
- View Event Listeners

In this step, complete the implementation of the app by setting up event listeners and handlers, and implementing persistence of the data to the device file system. You'll learn how to use other features such as Toasts and logging as well.

- Objectives
 - Wiring code and design
 - Using `onClick()` handlers
 - Setting up event listeners
 - Understand functional areas
 - Persistence
 - Toasts and Logging

3. Submitting your App

3.1 - Submitting via GitHub

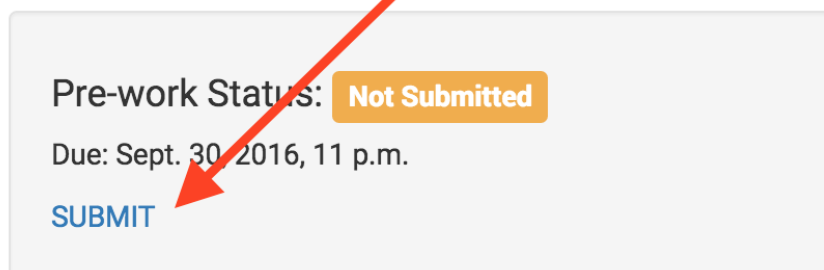
- 📁 Submitting a project via GitHub (27:38)
- README template
- Collaborating with Git

All submissions happen via Github and we require each submission to follow a particular format, including a **clearly documented README** with a list of completed stories and a GIF app walkthrough. Once you have completed the ToDo application, pushed your code to GitHub with the README and GIF walkthrough included, you'll submit the project.

3.2 Updating Your Application

Once you have **completed all required user stories and added a README as described above** then you are ready to notify us that you are ready for a pre-work review. To do this, go to the application status dashboard and then press the "SUBMIT" button in the pre-work section:

You should [begin working on the pre-work project](#) and submit to us below once completed.

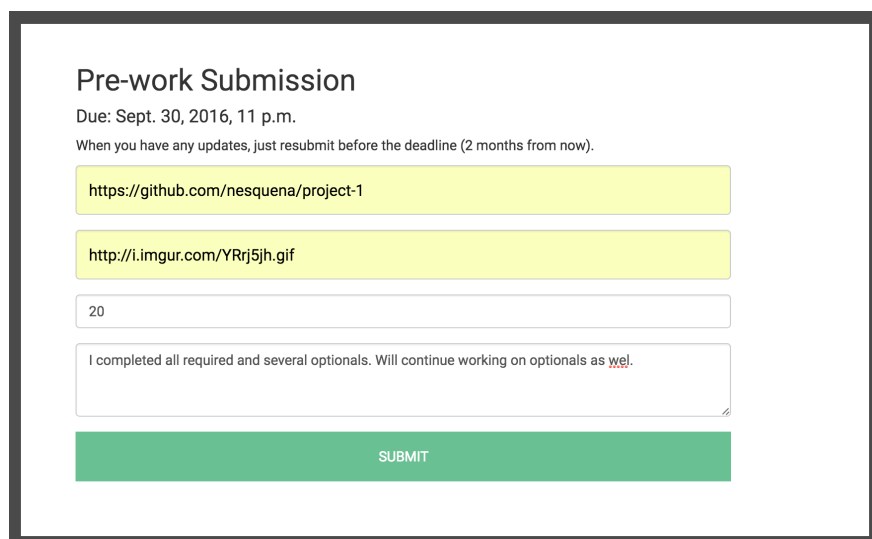


Pre-work Status: **Not Submitted**

Due: Sept. 30, 2016, 11 p.m.

[SUBMIT](#)

Make sure that this is the correct cohort before submitting. In the page that appears, enter the required information about your project:



Pre-work Submission

Due: Sept. 30, 2016, 11 p.m.

When you have any updates, just resubmit before the deadline (2 months from now).

[SUBMIT](#)

Then press "Submit" at the bottom to push your submission. Note that **you can always update your submission at any time** in case you want to re-upload the GIF or update the hours spent.

Note: All project code should be stored on GitHub. If needed, review how to push your apps to GitHub with this handy tutorial. You should add a .gitignore to your project as well to **avoid storing compiled and local files in git**.

Windows Users: For LiceCap to record GIFs properly in certain versions of Windows, be sure to set Scaling to 100%.

3.3 - Submission Checklist

Please **review the following checklist** to ensure your submission is valid:

- Can you **successfully add and remove items** from the todo list within your app?
- Does your app **persist todo items** and retrieve them properly on app restart?
- Did you successfully **push your code to github**? Can you see the code on github?
- Did you **add a README.md** to the repo on github copied from this template?
- Does your README include a **GIF walkthrough** of the app's functionality?
- Does your README include your **responses to the project analysis questions**?
- Did you visit your application dashboard and **submit using the pre-work form**?

If the answer to all of these is **YES** then you've successfully completed the basic todo app. When you've completed this, we will schedule a short phone call with you so we can chat further and answer any questions you have about the bootcamp and the rest of the selection process. In the meantime, you should focus on **improving the functionality and user interface** of your Todo app as outlined below.

4. Adding Edit Item Functionality (Optional)

4.1 Adding the Editing Feature

- 🎬 SimpleTodo Stretch Goals (23:44)
- Using Intents to Launch Activities

In this **optional** step, we will implement a new feature to allow users to **edit the text of an item added to the list**. For this task, the user should be able to **tap a todo item in the list and bring up an edit screen for the todo item** and then have any changes to the text reflected in the todo list.

Objectives:

- Adding a new feature
 - Additional Activity
 - Using Intents and Extras

4.2 Re-submitting your Changes

Once you've completed this feature, be sure to **checkin your changes to Github** and **re-submit these updates** to us by visiting the application status dashboard and then press the "SUBMIT" button in the pre-work section.

5. Extending Your Todo App Further (Advanced)

After initial submission, you may choose to iterate on the app by adding several advanced features to your todo app. If you are interested in pushing your knowledge, try your hand at implementing the following user stories and any other extensions of your own choosing:

- **(Suggested)** Improve the UI / UX of your app including icons, styling, color, spacing of your app.
- **(Advanced)** Improve style of the todo items in the list using a custom adapter
- **(Advanced)** Add support for completion due dates for todo items (and display within listview item)
- **(Advanced)** Add support for selecting the priority of each todo item (and display in listview item)
- Anything else that you can get done to improve the app functionality or user experience!

In addition to adding features and UI polish, be sure to check out how to improve the organization of your code and meet the Android style guidelines by reviewing our guide on how to organize your Android app files and folders.

Be sure to include in the README an updated GIF walkthrough using LICECAP of the app demoing how it works with required user stories completed.

Troubleshooting

Running into issues with your app, such as crashes? Check out our solving problems in Android apps guide to learn about how to debug and correct issues as they come up. You can also reference this list of common issues specific to this assignment.

A few other important troubleshooting tips are below:

1. **Hello World app doesn't run** - If the "Hello World" app that you loaded during setup doesn't run for any reason, you can skip that step and move directly to the todo app which will have you generate a new project.
2. **Emulator Booting** - Make sure to install HAXM and setup the HAXM emulator if you are on an intel-based machine as explained in the slides. Also on Android, make sure to **boot up the emulator first** through the Virtual Device Manager and verify that the virtual device is running properly **before attempting to run the app**. Also, do not close the emulator each time you want to run the app. **Leave the emulator open** and when you run the app will be re-installed onto the existing virtual device.

Quick Jump

- Android Pre-work: Todo App
 - 1. Setup Android
 - 2. Build the SimpleTodo App
 - 3. Submitting your App
 - 4. Adding Edit Item Functionality (Optional)
 - 5. Extending Your Todo App Further (Advanced)
 - Troubleshooting