Name: Sukhmeet Singh Hora

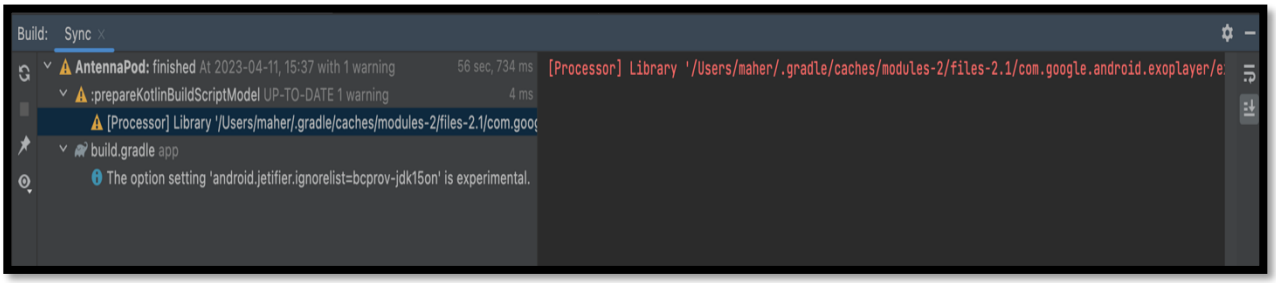
Student ID: 7884859

COMP 3350 Assignment

Legacy Software Assigned: Antenna Pod

The legacy software that you're being assigned (to be read as: stuck with) is Antenna Pod, an open-source podcast player for Android.
Cloned the project https://github.com/AntennaPod/AntennaPod.git in Android Studio.

1. **Build the system from the source.**



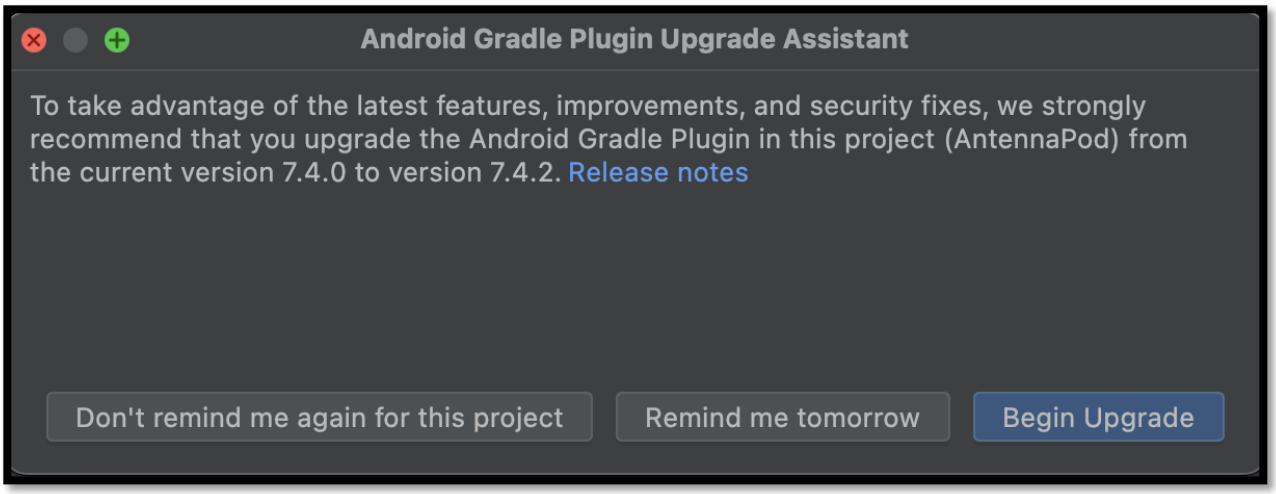
ANS 1,2 :

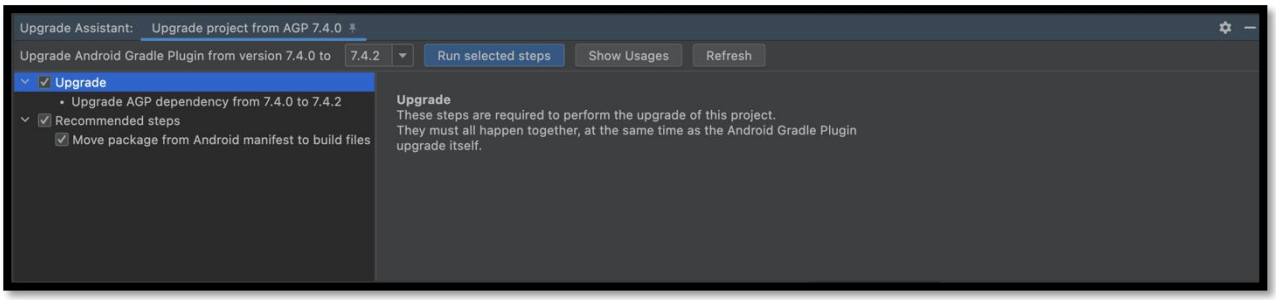
Got a warning [Processor] Library '/Users/maher/.gradle/caches/modules-2/files-2.1/com.google.android.exoplayer/exoplayer-ui/2.14.2/17c6bba9037fd8af7b69ed4a20af661ace85bd6d/exoplayer-ui-2.14.2.aar' contains references to both AndroidX and old support library. This seems like the library is partially migrated. Jetifier will try to rewrite the library anyway which was asking me to download a library, Jetifier will anyway rewrite this library.

Build. Gradle The option setting 'android. jetifier. ignorelist=bcprov-jdk15on' is experimental. Affected Modules: app

Starting Gradle Daemon...
Gradle Daemon started in 597 ms
> Task :prepareKotlinBuildScriptModel UP-TO-DATE

BUILD SUCCESSFUL in 8s





BUILD THE SOFTWARE: OUTPUT

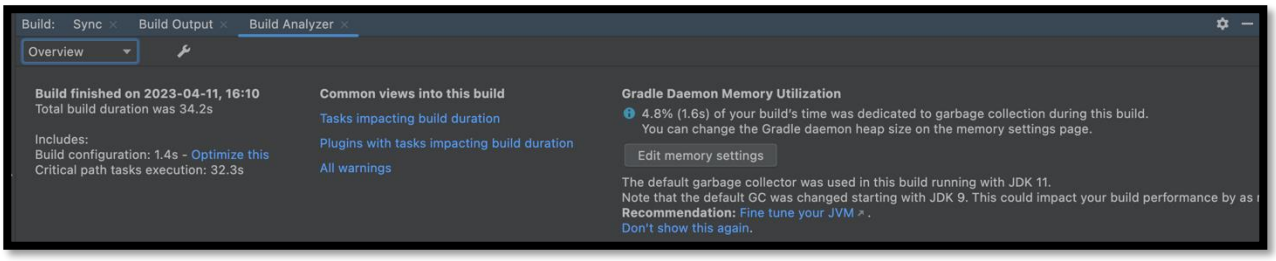
Q2. Describe the process of building the system. Were there any difficulties (compatibility, dependencies, etc)? Did you have to modify anything, or find additional libraries?

There were some warnings like:

- warning: [deprecation] various functions in the fragment have been deprecated.
- warning: [unchecked] unchecked generic array creation for varargs parameter of type Transformation<Bitmap> []
- warning: [rawtypes] found raw type: Future.
  Observable.fromCallable((Callable<Future>) () -
  DBWriter.removeFeedNewFlag(selectedFeed.getId()))
  ^ missing type arguments for generic class Future<V> where V is a type-variable: V extends Object declared in interface Future.

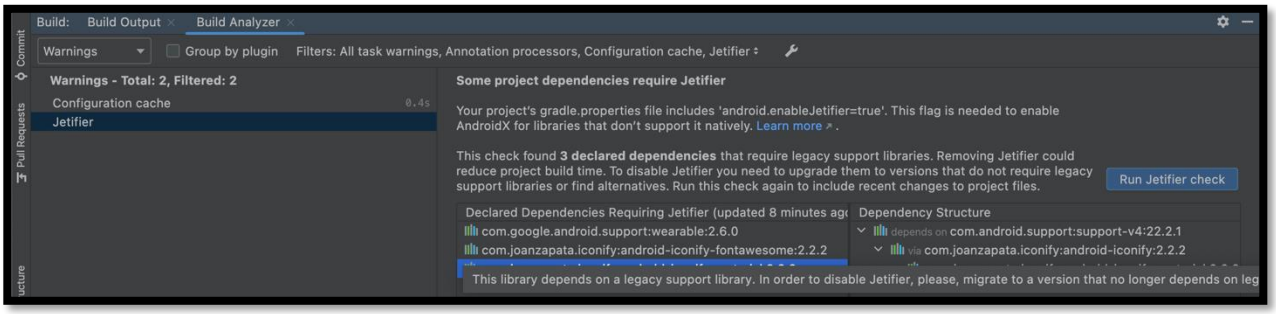
- The option setting 'android. jetifier. ignorelist=bcprov-jdk15on' is experimental.

BUILD SUCCESSFUL in 34s
455 actionable tasks: 455 executed
Results from Built Analyzer.



Checked how to optimize the Build Configuration

There were some declared dependencies that require Jetifier, need to upgrade them to versions that don't require legacy support libraries, and this would reduce project build time.



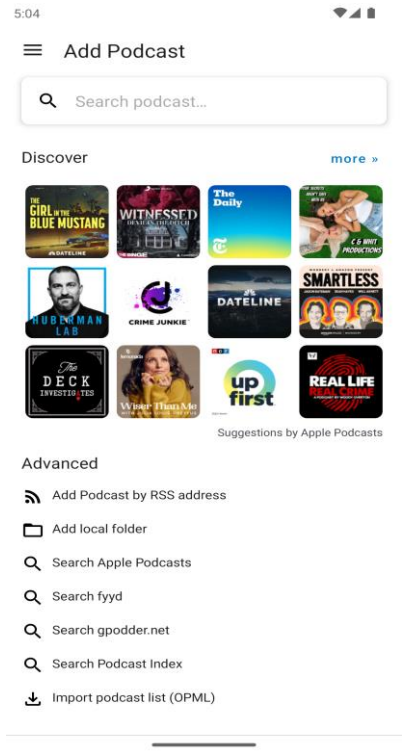
As the build was successful, move to the next steps.
BUILD SUCCESSFUL in 34s
455 actionable tasks: 455 executed

Q3. Explore the system to get a feel for what it does. As your response for this activity, briefly describe the **purpose** of the system, and include a screenshot.

Purpose of Antenna Pod: Allow the users to manage podcast easily by playing podcasts and listen form anywhere , adding new podcasts by searching or importing or looking at suggestions, downloading, view playback history to view a previously listened podcast, subscript

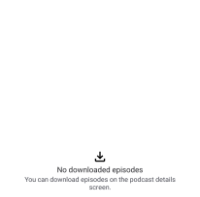
to new podcasts and add subscriptions, view added podcasts, receive inbox messages for newly added episodes of the podcasts, add podcasts to queue and in process of have setting that allows users to change appearance, playback setting, storage and other settings.

- Adding new podcasts:



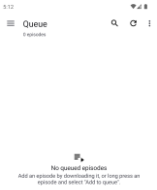
- Playback History:

- View Downloads:



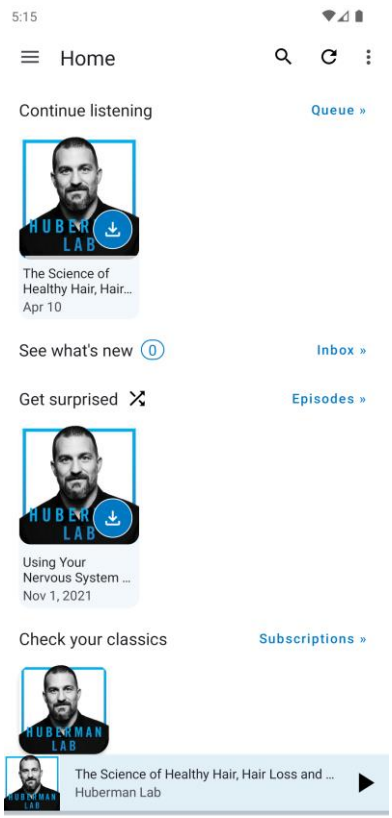
- Subscriptions:

- View Added Episodes:



- Add podcasts to queue:

- View inbox messages for newly added episodes:



- Home: This Home Screen view the podcasts added, manages queues, check new episodes (get notifies if allowed on asking), checks subscriptions and classics.

Q4. Explore the source code for the software. Is there a discernible **architecture**? If an architecture diagram exists, comment on it. Is the architecture diagram sufficient for you to understand where code in the software lives?

Ans4.
After exploring the source code, it can be seen or observed that a specific architecture has been followed. The whole source code has been divided into various layers/Java Modules.

- App
- Core
- Event
- Model
- Net
- Parser
- Playback
- UI

Though there is no architecture diagram available to understand where the code for the specific feature lives in the software, from the different Java modules it can be quite easy to understand where the code by just looking at the java classes in each module, it is quite understandable from the name itself. The layers are not defined but there are different Java Modules for different functionalities and layers.
The code is well divided into presentation layer which handles user interface, different adapters and fragments, logic layer which handles the complex logic like preferences, backup, sync, storage and database layer with various Java Modules like storage with database.

Q5. Choose **two** distinctly different maintenance tasks for the system: one new feature and one bug fix. You **will not** actually change the code. Describe the tasks that you want to accomplish in high-level terms.

Ans5:

BUG:

- **Podcast Specific Playback Speed doesn't work:**
  - Add 2 podcasts to the subscription list.
  - Open a podcast episode to listen on one podcast and set its play back speed to 3.00 listen for a bit.
  - Bo back and open another episode of the other podcast, that podcast will also be played at play back speed of 3.00.
  - Though the expected behaviour is to reset the playback speed to 1.0 if the episode played is of another podcast. In this way, every podcast has can be set to a different/specific play back speed.
- **Steps and ways this can be fixed:**
  - Have a global variable "playbackSpeed" that maintains the currently set play back speed.
  - Whenever the playback has been changed by the user, change the value of the global variable (to save the play back speed for currently running podcast).
  - The value of the global variable must be checked/changed/ reset whenever a new episode is played:
    - If episode is of the same podcast: play it at the current set play back speed (value saved in the global variable) and then set the playback speed attribute of the episode.

        - If the episode of a different podcast: reset the playback speed (global variable to 1.0 and set the attribute of episode).

ONE NEW FEATURE:

- **Allow the users to upload a podcast for public and became a creator to contribute to the podcast library.**
    - **User story #1**
        - **As a user I should be create creator account and login**
        - **As a user I should be able to create content by uploading personal podcasts.**
        - **As a user I should be to contribute to the public video library.**
    - Introduce a login feature to different between normal user account and creator account.
        - Creator accounts must be verified to account for authenticate and clean background checks.
        - Creators must be rewarded for each of their uploads depending on the times their podcast is heard.
    - Add a button on account to upload podcast and make it available to eb viewed publicly.
- **Ways this feature can be implemented:**
    - **Login:** allow users to create accounts and decide if they want to be a normal user or creator.
        - Have verification steps if creator account created and verify if all verifications passed and give upload access to that account.
        - Have database that saves account information like username, password, account type and library.
    - **Upload a podcast** only for verified creator accounts:
        - When podcast uploaded, add it to the public library and make it available to be visible to all users.

Q6. For each of the two tasks, locate and record where you would insert a **seam**. Describe how you would refactor this code to add the seam. What kind of **tests** would you need to implement for that component?

BUG:

INSERT SEAM:

- PlaybackServiceMediaPlayer.java in https://github.com/AntennaPod/AntennaPod/blob/39d309e906c0b0b3a199064505ce76bc8045abd9/playback/base/src/main/java/de/danoeh/antennapod/playback/base/PlaybackServiceMediaPlayer.java
  - It already has methods to setPlaybackParams () in which speed is a float parameter and getPlaybackSpeed ()
  - It also has functions to get selectedAudioTrack.
- SpeedChangedEvent.java in https://github.com/AntennaPod/AntennaPod/blob/39d309e906c0b0b3a199064505ce76bc8045abd9/event/src/main/java/de/danoeh/antennapod/event/playback/SpeedChangedEvent.java
  - Has methods to get new Speed and constructor sets the new speed.
- Addition/Refactoring: Whenever new episode is clicked,
  - If the playback speed is changed, create a new speedChangedEvent() for the podcast and then call setPlaybackParams.
  - Whenever new podcast is played then call setPlaybackParams with the default playback speed.

TESTS:

- **Unit tests:**
  - **Test1**
    - Play an episode of any podcast.
    - Change the playback speed to 3.0.
    - Play another episode of any other podcast.
    - Assert that playback speed of the second episode is 1.0(default).
    - Assert the playback speed of the first episode is 3.0.
  - **Test2**
    - Play an episode of any podcast.

- Change the playback speed to 2.5.
- Play another episode of same podcast.
- Asset that playback speed is 2.5.

FEATURE:

INSERT SEAM:

- For creating an account introduce create and login layouts in UI java module https://github.com/AntennaPod/AntennaPod/blob/39d309e906c0b0b3a199064505ce76bc8045abd9/ui/common
- In the Core module, add verification logic to verify creator account, once verified give upload access by showing the button https://github.com/AntennaPod/AntennaPod/blob/39d309e906c0b0b3a199064505ce76bc8045abd9/core/src/main/java/de/danoeh/antennapod/core
- Create button to allow only creator user to upload a podcast.
- Database in Storage module https://github.com/AntennaPod/AntennaPod/blob/39d309e906c0b0b3a199064505ce76bc8045abd9/storage/database store the account created with preference as creator or user as selected.
- Once podcast is uploaded successfully, add the podcast to database storage library such that it can be seen by all users, in https://github.com/AntennaPod/AntennaPod/blob/39d309e906c0b0b3a199064505ce76bc8045abd9/storage/database/src/main/java/de/danoeh/antennapod/storage/database/mapper

TESTS:

- **Unit Tests:**
  - **Account**
    - Create a new account successfully test Username test and log in with the same account to test successful login.
  - **Upload podcast**
    - Click on the button to upload podcast.
- **Integration Test**
  - **Account with Database**
    - Test account created successfully with setup where username and pass are saved in database and

preference as creator, testLoginSuccess and assert the expected username and account Type.

- **Podcast upload in library database**
  - upload the document and save that in database, assert that videolibrary contains the uploaded video.

- **<u>Acceptance Test</u>**
  - **To test the overall system where account is created, saved and then video is uploaded in library and asserted that it is visible to public.**
    - Create a new account with creator type account.
    - Verify the account (and expect to have creator access).
    - Log in to account and expect to see welcome screen and upload podcast button (asserting creator account).
    - Click on the upload button and select a valid podcast to upload.
    - Expect to see message that video is uploaded.
    - Click on the public video library and confirm the video is uploaded.
    - Check if the account to other users (confirming it is added to app database).

Q7. Briefly describe the maintainability **of this source code. Are there any aspects of the project (architecture, design, code, etc) that make it particularly easy or difficult to modify?**

- Design Principles followed.
  - There are separate modules and singleton classes whose resources are shared, once understood the overall structure, the code can be added for new features/refactored to fix bugs easily. The code is low coupled and has high cohesion. This makes easy to have changes.
- But the code uses some dependencies which increase the build speed and are not updates, many functions have deprecated code that must be changed, have raw types of warnings which missing argument type and the layers must be mentioned in architecture diagram explicitly or in form of packages just to be easy for the contributors.

Reference Used:

1. Antenna pod GitHub Project https://github.com/AntennaPod/AntennaPod
2. Issues- antenna pod Project https://github.com/AntennaPod/AntennaPod/issues
3. Antenna pod Website https://antennapod.org/