[[1]](#footnote-2)

*Overview*— This project was to create a to do list Android application that would allow users to add, delete, read, and update items in a to-do list while also scheduling notifications for those to-dos. This assignment was done by implementing a level of separation of concerns with maximum usability in mind. The result was a very usable to-do-list that allows of the required functionality except for storing changes to a file when offline and updating the db when connection is restored.

Assignment 2 – To-Do List

Luke Brandon

# INTRODUCTION

This project is to design and develop and Android application that is a functioning To-Do List. This app should allow the user to add, delete, read, and update the todos that are on their to-do list. The application consists of 2 activities, one for viewing the entire to-do list and another for viewing and/or editing the contents of a to-do. The application should store all of the information for all of the to-dos in a SQLite database on the phone.

# Application Design

When choosing a design methodology, I tend to lean towards MVP for its simplicity and separation of concerns, that was the idea for this project. There is a distinct model that access the SQLite database along with various files for the different views. Because of the simplicity of the project, the View and Presenter are effectively combined into the activities however, given more time and more complexity, having separate views and presenters would make development far simpler. The model is a SQLite Database that stores the title, description, due date, and completion status of each item in the to-do list. The activities ask the ToDoProvider to get the information from the database and also tell the ToDoProvider what and when to update, add, or delete to the DB. There are also utility classes that are used to get the connection status of the device as well as the scheduled notifications which, when clicked, launch the application into the to-do list item detailed view that would be expected with an artificial back-stack created.

## Flow

When designing this application, there were multiple development decisions that were made to make the experience of the user better a few of these include:

- Default due date to current day if not specified

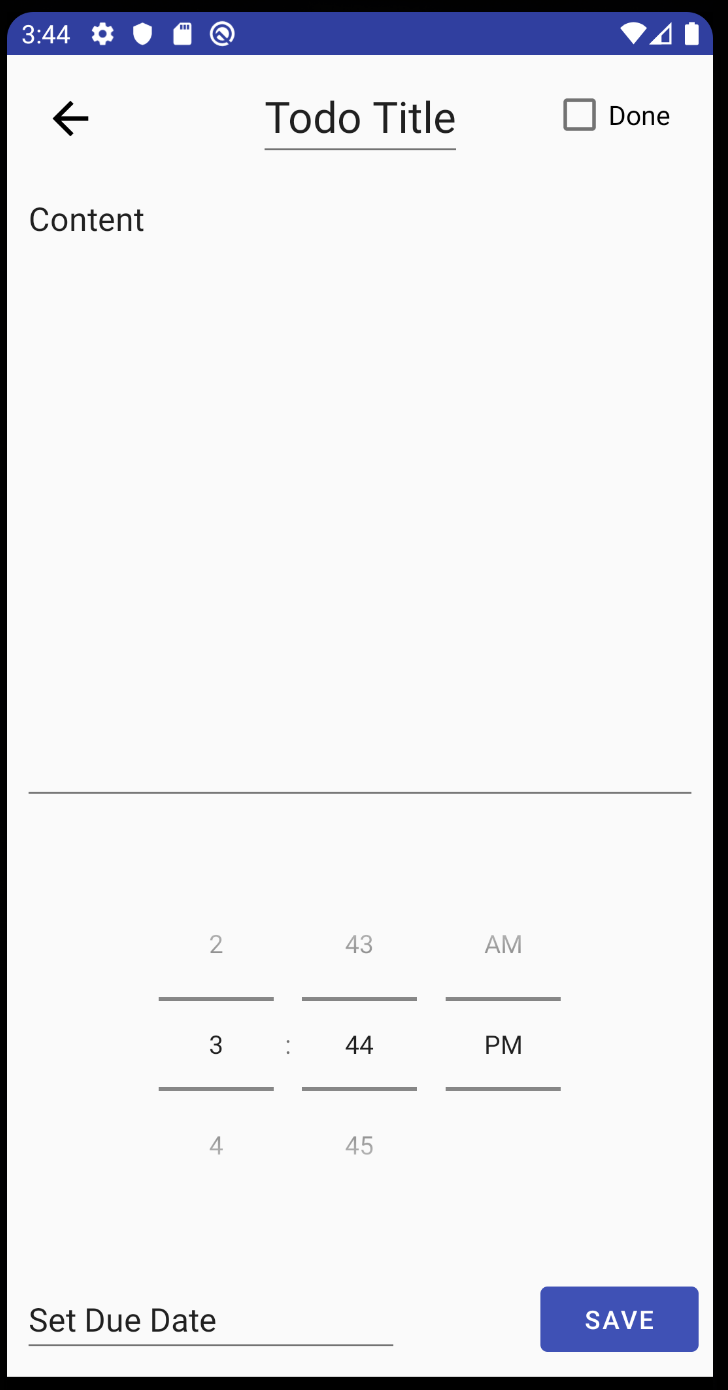
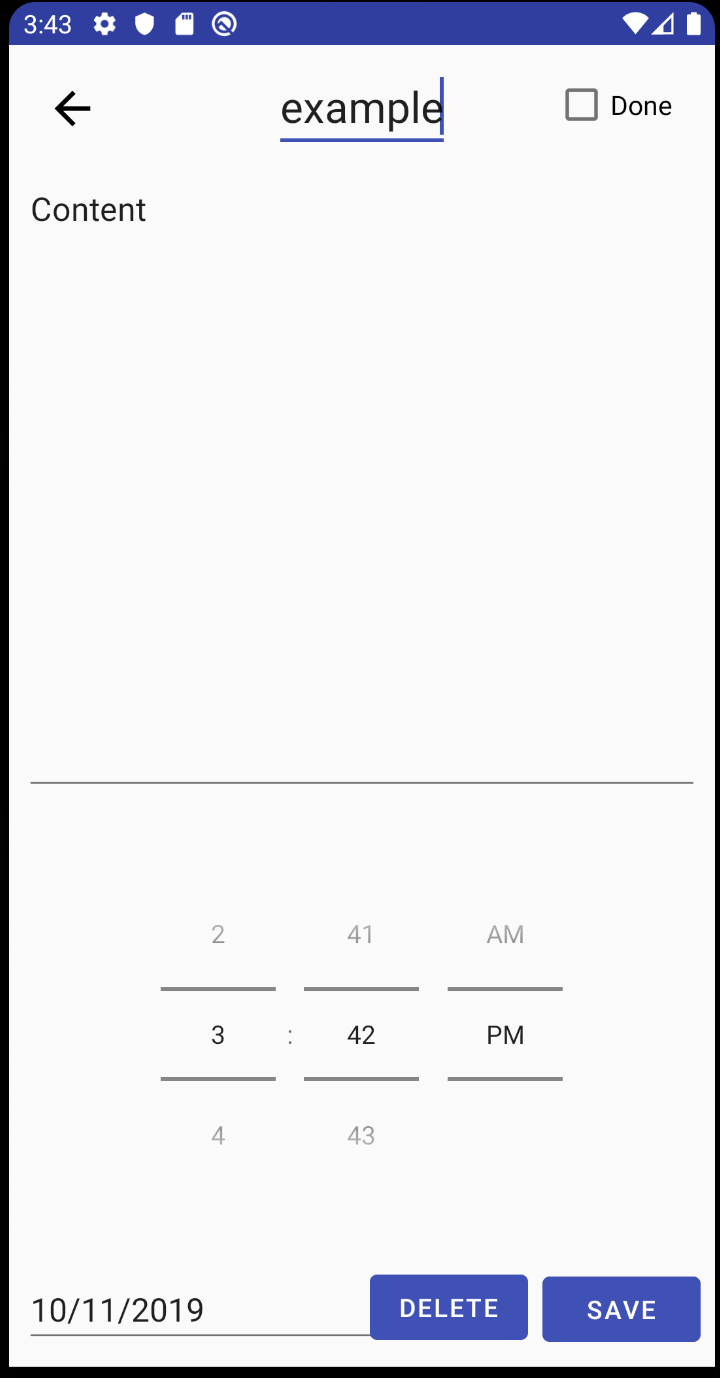
- Real-time due date display updating

- When creating a new to-do the delete button is not present as there is nothing to delete

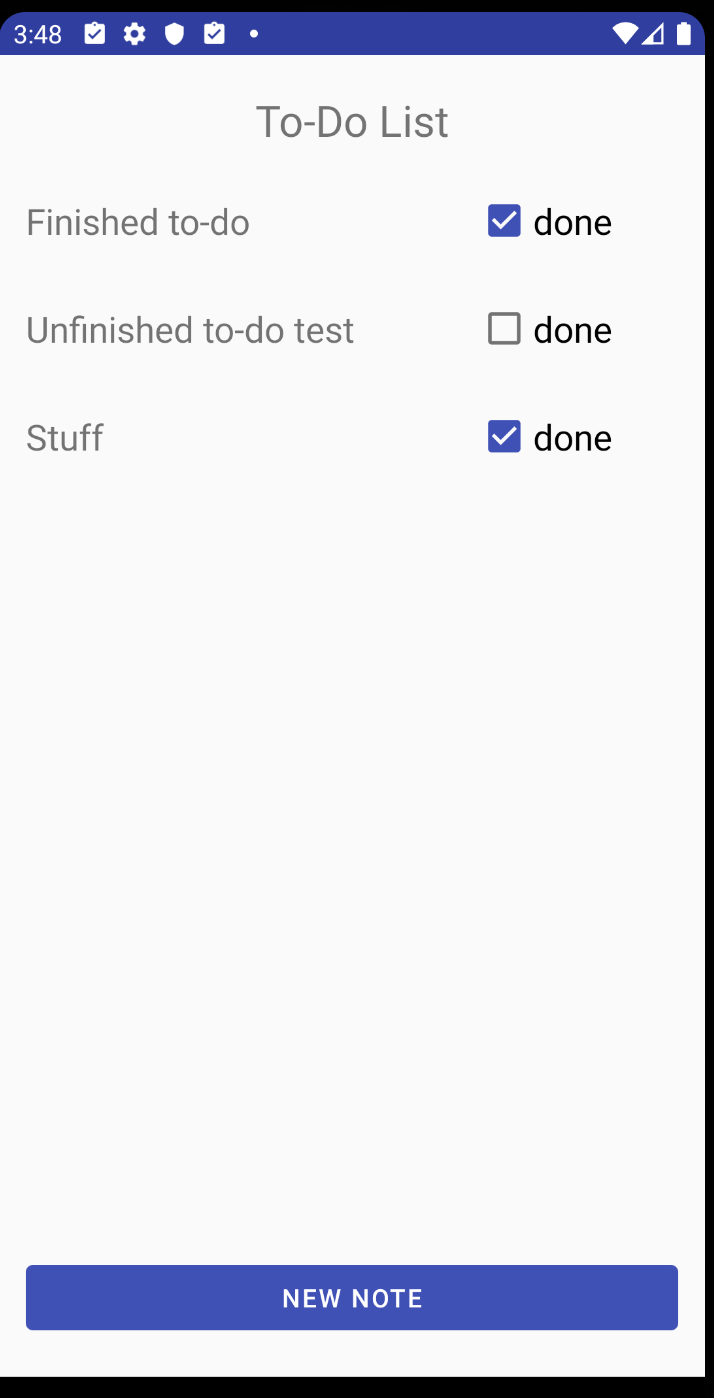
- Scheduled notifications are canceled when a to-do is deleted and are canceled and updated if the due date has changed on the to-do.

# Results

The application performs well and completes all of the necessary functionality except for writing to a file when internet connection is lost and then updating the database when the internet connection is restored.



The image on the right is an example of the view when creating a new note, there is not delete button because there is nothing to delete since it has not yet been created, the image on the left is the same activity but was started by clicking on a to-do in the to-do list, the delete button I present because it can actually be deleted.



The above image shows the main screen of the to-do list, where the user can see all of their to-dos, update their status, or click on them to edit them further or delete them.

# Future Work

The code architecture for this application is less than optimal. From the start I should have decided on a design methodology like MVP and developed the app on top of that. I did my best to abstract out the different aspects like views and the model, but currently a lot of the log is done in the activities when it should be done in presenters that present the information that needs to be displayed to the views. With the additional complexity that not having a rigid design structure added, in the future I will decide on and develop under the MVP methodology from the start to avoid the additional complexity and difficulty of figuring out problems. If this application were to be developed further, the MVP methodology would make the code scale significantly more simply, and efficiently.

1. [↑](#footnote-ref-2)