GeoNotes Splice Extension

Client: Trimble

Members: Jackson Will, Elijah Mt. Castle, Connor Bilgrave, and John Simpson

6/15/21

Introduction

Why are we doing this?

When operating a tractor, it would be beneficial for operators to have the ability to mark different hazards and other valuable information throughout their fields. A system used for: avoidance of a ditch or boulder; providing general information about the crops in a specific field; or marking areas where pest control is needed and would be invaluable to operators. In addition, analyzing information from the U.S. Department of Agriculture, the National Agricultural Safety Database states that tractors are responsible for between ⅓ and ½ of deaths and disabling injuries for farmers, and ½ of these are from rollovers. GeoNotes indicating dangerous terrain that could help prevent these tragedies.

What have we done?

Using the Splice API provided by Trimble we created a GeoNote extension to add onto the existing Precision IQ android application currently used in tractors. This extension gives the user the ability to place GeoNotes at points of interest. When a GeoNote icon is clicked it places a note at the current location and when the camera icon is clicked it allows the user to assign a picture to the last created GeoNote. Created GeoNotes can then be viewed with their geographic latitudinal and longitudinal location on a map and in a list format. In addition, when a list item is clicked, the details of a specific GeoNote can be viewed with valuable information about the surrounding environment including the type, location, and images. In addition, the framework for adding audio files to a GeoNote was implemented to provide future developers with the resources to implement the feature.

Requirements

Functional Requirements

- GeoNote
    - GeoNotes have saved GPS locations, and display that location on the map.
    - GeoNotes have a type: trash, hazard, product, animal, equipment, spill, and water hazard.
    - A GeoNote has the option for users to include associated images taken from available cameras, and the framework for audio clip descriptions taken from an attached microphone.
- User Interface
    - Selecting a GeoNote on the map opens an Extension page with information about the specific GeoNote, such as type, creation date, associated pictures, and a placeholder for the audio recordings
    - Selecting the icon, located in the GeoNote Drawer creates a new note of the specific type.
- Room Database
    - GeoNotes are stored with necessary information such as Type, Location, Time of Creation, and the path to associated pictures and recordings.
- The project must run on Android Marshmallow
- The project must run alongside the existing Precision IQ software.

Non-Functional Requirements

- A GeoNote can be created with the minimal number of clicks while operating the tractor
- Any UI features added should follow the style of the existing UI
- Match existing design language
- The extension must be written in Android Studio
- Through the stages of the project, team members have articulated thoughts for improvement and reported possible bugs in the Splice API.
- The extension correctly interfaces with the Splice API

System Architecture

UI Designs

Shown in the background of **Fig. 1** is the runscreen from the existing Precision IQ application. Users can select the book icons in the navigation bars to navigate to different aspects of the GeoNotes splice extension. The left book icon navigates to the GeoNoteExtension page shown in **Fig. 2** and the right book icon opens the GeoNote Drawer shown in **Fig. 3**.

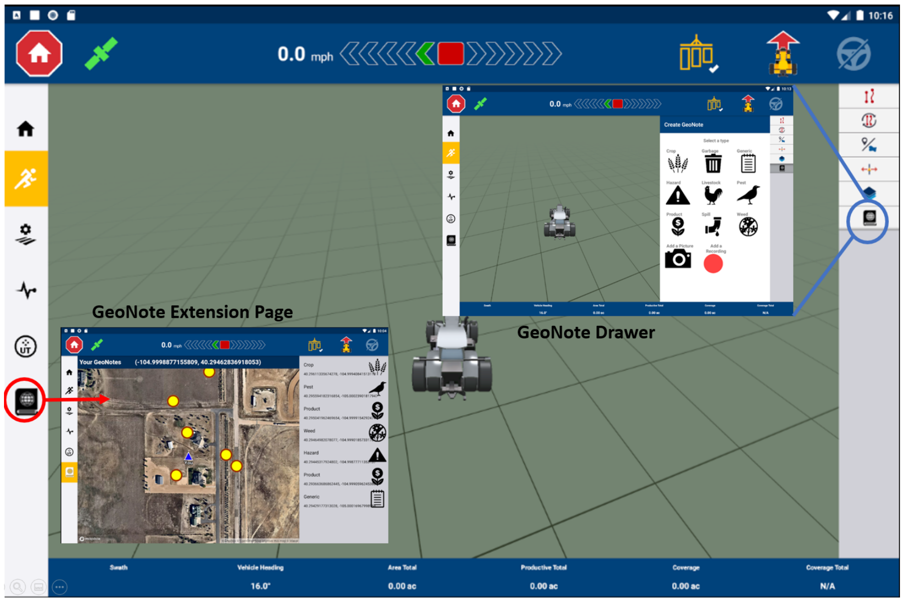
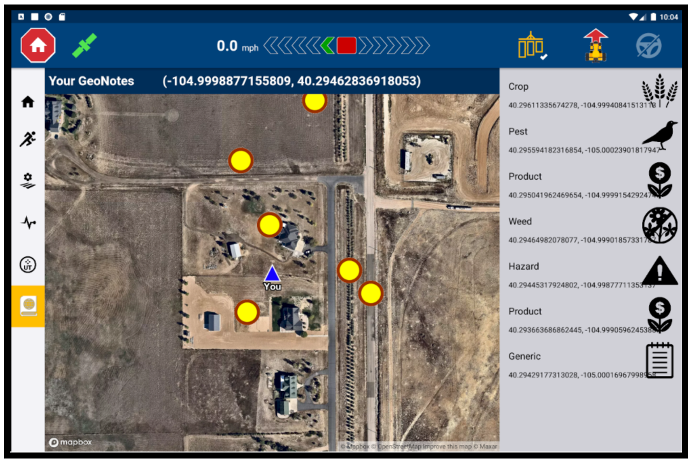


Figure 1 UI Flow

This displays the navigation between content within the GeoNote extension.



Your GeoNotes (-104.9998877155809, 40.29462836918053)

Crop
40.29611335674278, -104.99940841513118

Pest
40.295594182316854, -105.00023901817947

Product
40.295041962469654, -104.99991542924

Weed
40.29464982078077, -104.99901857331

Hazard
40.29445317924802, -104.99877711353

Product
40.293663686862445, -104.99905962453

Generic
40.29429177313028, -105.00016967998

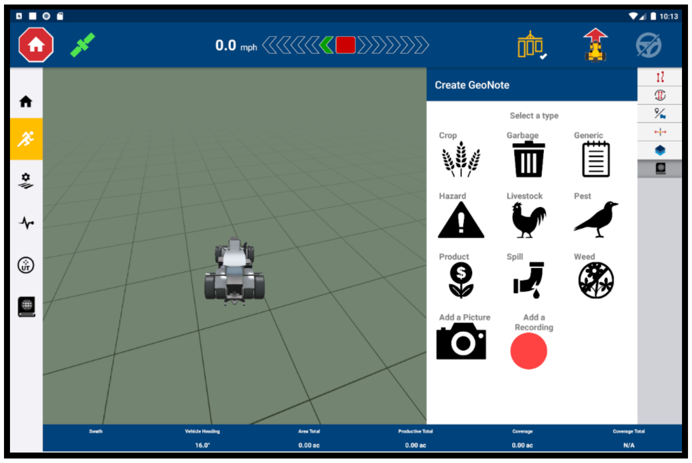
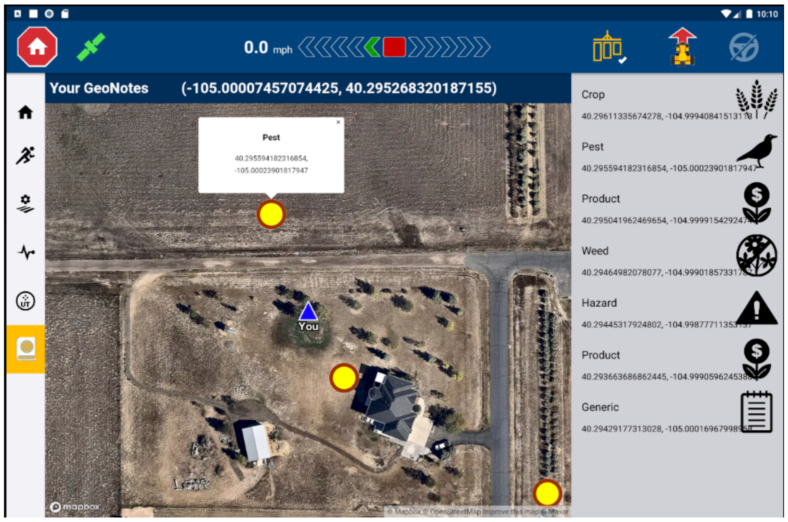


Figure 2 GeoNote Extension Page
This is where GeoNotes can be reviewed on the map and through the detail page. Detailed functionality is shown in **Fig. 4** and **Fig. 5**. Note the left bar has the GeoNote icon highlighted, this denotes that the Extension Page is the current page.

Figure 3 GeoNote Creation Drawer
This drawer is used to create new GeoNotes of the specified type and a picture can be added to the last created GeoNote. The recording functionality framework is also in place. Note the right bar has the GeoNote icon darkened, this illustrates that the drawer is active.



Your GeoNotes (-105.00007457074425, 40.295268320187155)

Pest
40.295594182316854,
-105.00023901817947

Crop
40.29611335674278, -104.99940841513118

Pest
40.295594182316854, -105.00023901817947

Product
40.295041962469654, -104.99991542924

Weed
40.29464982078077, -104.99901857331

Hazard
40.29445317924802, -104.99877711353

Product
40.293663686862445, -104.99905962453

Generic
40.29429177313028, -105.00016967998

Figure 4 GeoNote Extension Page details
On the left portion of the screen, a Mapbox map with all GeoNotes marked by yellow circles is displayed. GeoNotes marked on the map can be clicked to view their type and latitudinal and longitudinal location in a popup box. In addition, a list with all existing GeoNotes is displayed on the right side of the screen to reference the GeoNotes that are displayed on the map. Clicking a GeoNote in the list will open the GeoNote Details screen, shown in **Fig. 5**.

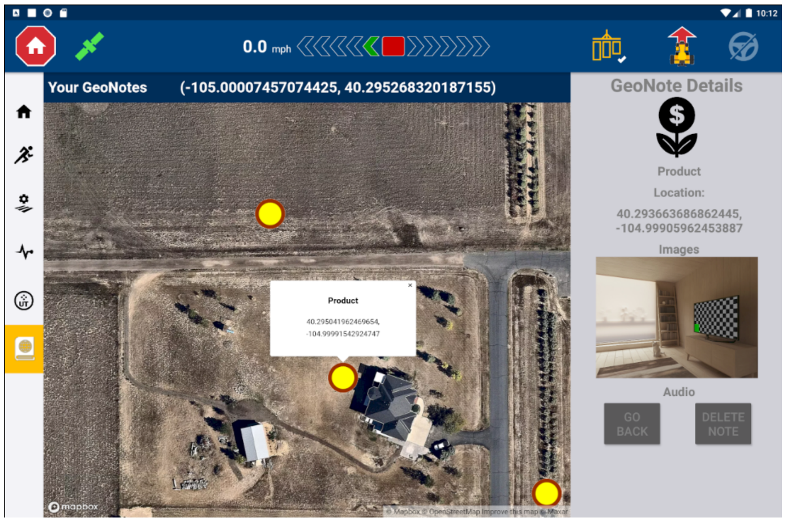


Figure 5 GeoNote Details screen
Inflated into the portion of the screen occupied by the GeoNote list, the details screen displays additional information about a specific GeoNote. All information, including the icon, type, location, image, and the future framework for audio files are visible here.

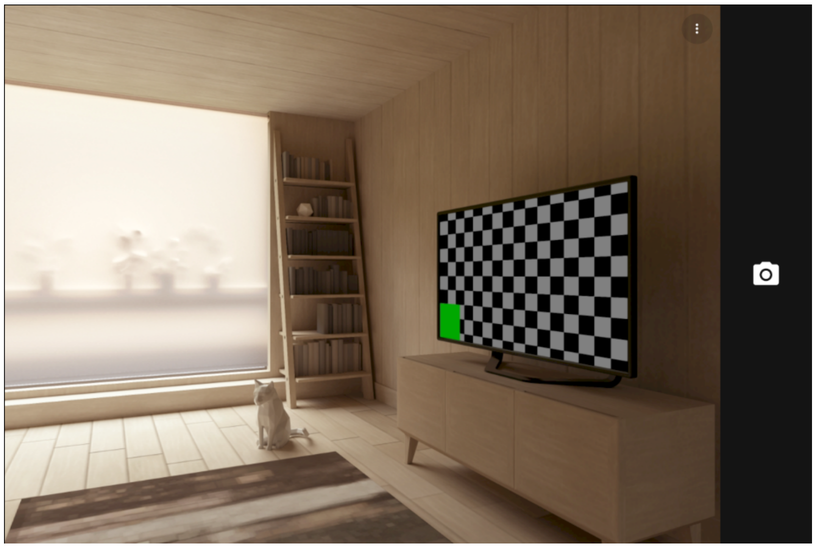


Figure 6 Camera screen
When the camera button is clicked, the user is redirected to the camera activity. Once the picture is taken and confirmed, the user is redirected back to Precision IQ and the picture is sent back to the app (this is an example image).

UML Diagrams:

The diagrams pictured, show the connections between the different packages of the GeoNote extension. In **Fig 7**, it shows the interactions within the package corresponding to the Geonotes Extension Page. In **Fig 8**, it shows the interactions within the package corresponding to the GeoNotes Drawer. In **Fig 9**, it shows the interactions corresponding to the Room database. All of the green boxes correspond to components provided in the source code, while all the blue boxes correspond to components that were added to the project.

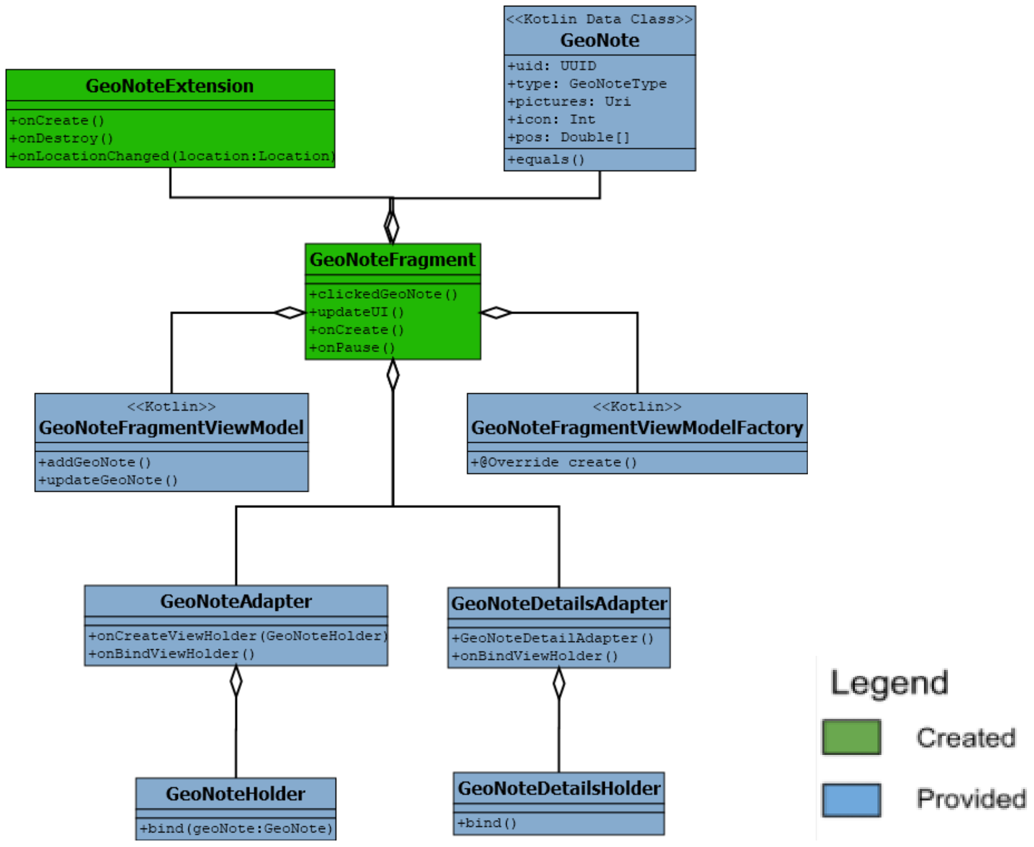


Figure 7 GeoNoteFragment UML diagram

This fragment represents the GeoNote Extension page, it provides the functionality to update and add GeoNotes to the map and list, and queries the database to inflate the detail screen.

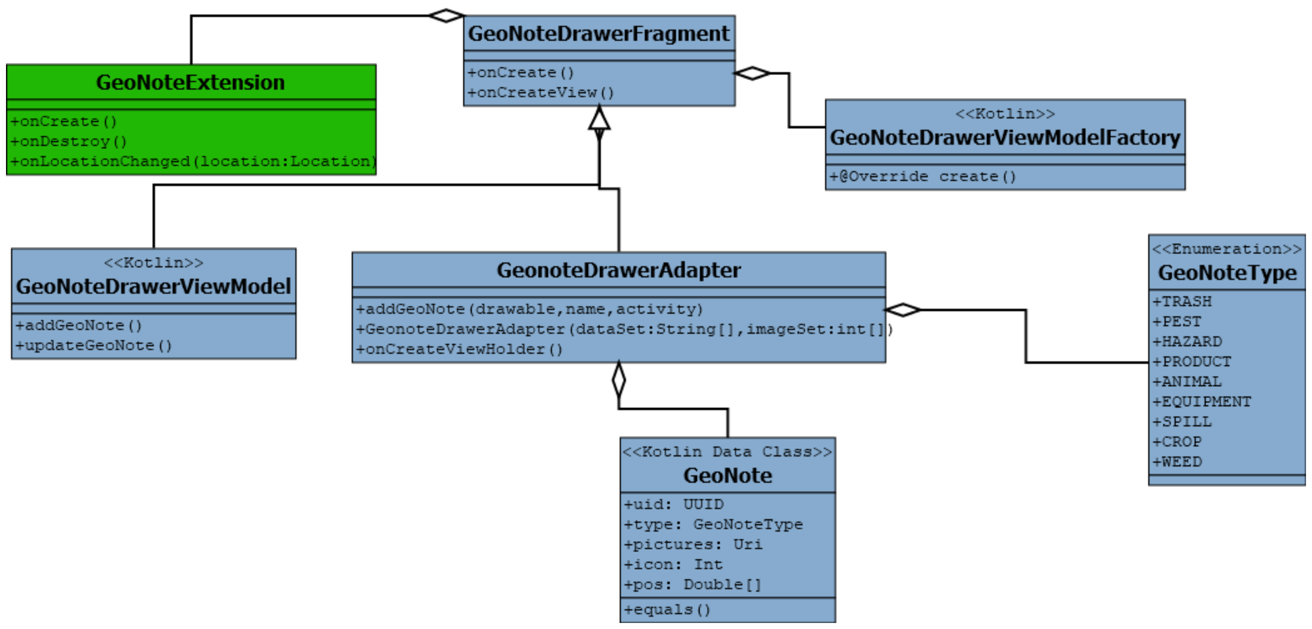


Figure 8 GeoNote Drawer Fragment UML diagram
This fragment represents the GeoNote Drawer, it provides functionality to display the different icons, take pictures, and to record audio. It also accesses the database to create new GeoNotes and add pictures to existing GeoNotes.

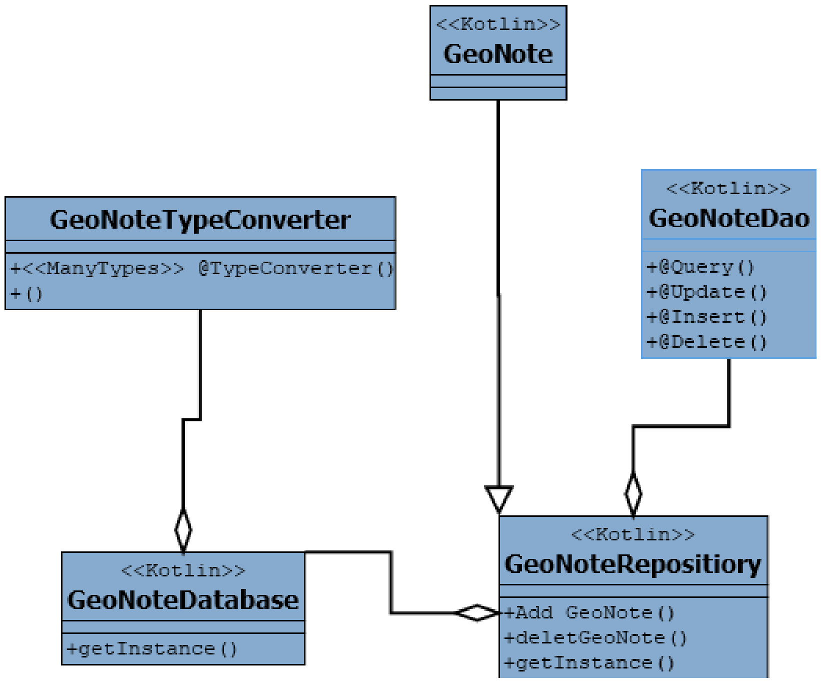


Figure 9: Database UML diagram
These classes are coded in Kotlin and provide the framework to manage the GeoNote database storage, queries, and updates.

Software Architecture:

Shown in **Fig 10**, it describes the architecture of the GeoNote extension. The Precision IQ application and the splice extension API have been provided. The GeoNote extension is constructed with three components. The GeoNotes Drawer is used to create new GeoNotes from a list of GeoNote types and add the created GeoNotes to the database. The database stores all necessary information locally about the GeoNotes in a concise format. The Extension Page is used to view existing GeoNotes on a map and to view information about existing GeoNotes.

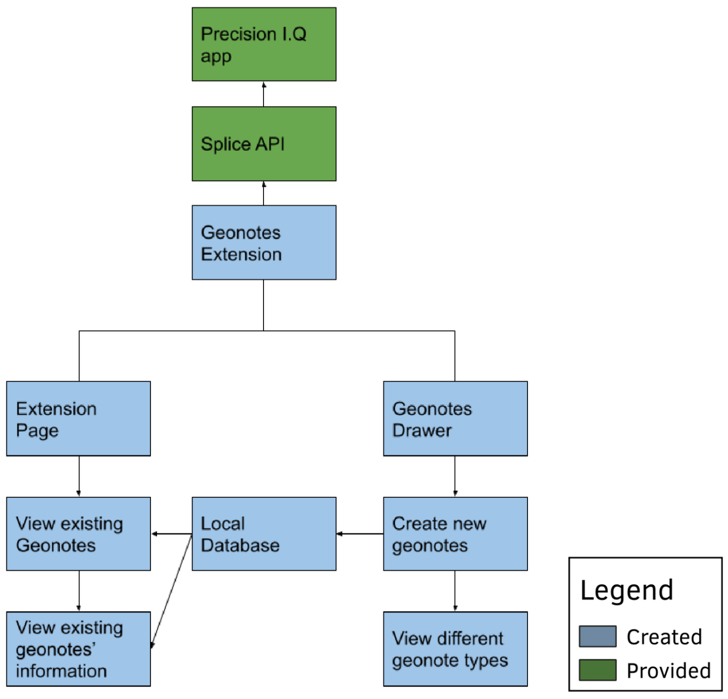


Figure 10 Software Architecture
This is a high-level overview of the project highlighting the important aspects of the software.

Room Database:

All GeoNotes are stored in the database according to **Fig. 11**. They are differentiated with a Java UUID, which ensures that each GeoNote is unique and no copies are added and no GeoNotes are overwritten. The information in a GeoNote is described along with its different data types in the table. In addition, type converter methods are used to ensure that all data is stored and queried correctly.

Database: GeoNotes	**Descriptions**:
ID (UUID)	Unique ID
Location (Double[])	GPS Location (Lat, Lon)
Audio File (Uri)	Optional Audio Clip
Photo File (Uri)	Optional Photos
Name (String)	Name of the GeoNote
Description (String)	Description of the GeoNote
Type (Enum)	Type of GeoNote
Icon (int)	The icon associated with the GeoNote

Database Structure **Figure 11**

Technical Design

Room Database

- This Android Studio feature streamlines the creation of a SQL database and simplifies the code to implement a database.
- The room database was implemented in Kotlin, which connected to the functional Java code to display and handle the data.
- The information was stored in strings, a Type Converter was used to convert the different information types back and forth from a string to the necessary data types for functionality.
- Live Data and Observers were used to monitor changes in the data to actively update the list and information within the application.
- Java's Universally Unique Identifier (UUID) was used to ensure that there are no copies of GeoNotes and existing GeoNotes are not overwritten.

MapBox

The mapping functionality is running in MapBox.

- MapBox is a web page overlay for Open Source Maps (OSM).
- The map was wrapped in Android's webview container to display a webpage in the application.
- The MapBox is running with a mix of HTML, CSS, and Javascript.
- The main project interfaces with Javascript functions to draw points and manipulate the map.

Web listeners were used to monitor the status of MapBox.

- Mapbox can not draw when loading and does not have its own functionality to draw later, so a weblistener was used to monitor when it was finished loading.
- The web listener monitors the different status messages of MapBox and then resends draw commands when the page is finished loading.

Camera and Audio

The GeoNotes extension uses the Android Media Store to capture pictures and audio.

- Each GeoNote's UUID is used to create a unique file name for pictures and audio storage.
- A Try/Catch statement was used to prevent crashing if the camera and audio peripherals cannot be found.
- When a picture or audio recording is captured, the last created GeoNote is updated. The URI corresponding to a picture or audio file is assigned to the GeoNote.

Splice API

The API interface that is used for interfacing between Precision IQ and Android.

- Splice is designed for Marshmallow
- Extending the GeoNote into the Precision IQ sandbox, Splice changes many file and navigation permissions that result in unique errors.
- Select Android features are difficult or impossible to implement alongside the Splice API.

Quality Assurance

To insure a quality product the following software plan was implemented:

1. **Unit testing**

   The JUnit library was implemented for testing the functionality of the back-end of the GeoNote Extension. The following aspects were tested with JUnit:

   - Creation of a functional GeoNote.
     - Programmatic and UI driven creation of a GeoNote
     - GeoNotes have the required functionality.
   - Successful navigation between Fragments of the application.
     - Navigating to the GeoNote Drawer(**Fig. 3**).
     - Navigating to the Extension Page(**Fig. 2**).
     - Navigating to the GeoNote Detail Page(**Fig. 4**).
   - Successful creation of the database
     - Ability to successfully read and write GeoNotes to the database.
     - GeoNotes in the database are viewable from the extension page.

2. **User interface testing**

   The user interface was tested from within the application itself to ensure it was easily navigable. The user interface was also tested by the client to ensure it was navigable by a non-developer.

3. **Integration testing**

   As each section of the project was completed, the functionality was tested as a whole to ensure the different aspects of the project worked together.

   - The GeoNote Extension as a whole was tested from within the Precision IQ application.
   - The GeoNote Drawer and Extension Page were tested in conjunction with the database to ensure a new GeoNote can be created and reviewed.

4. **Code reviews**

   The code was reviewed through peer programming to ensure that coding practices were upheld and that no bugs existed in the code.

   - Coding Practices
     - Maintainability
       - Ensuring classes interact in a logical fashion
       - Classes can be easily expanded
       - Classes interact with a minimal complexity
       - Classes are organized into packages with associated classes
     - Readability
       - Naming schemes were maintained throughout the source code and the source code is well commented. Long function calls were broken onto multiple lines.

5. **User acceptance testing**

   The code was provided to the client to make sure it can handle required tasks in real-world scenarios, according to specifications. The project was installed on a real tractor to assess its quality.

6. **Code metrics**

   Code metrics were assessed by measuring a number of aspects
   - The size of both the source and application code.
   - Cyclomatic Complexity: The code was optimised to include as few code flow paths as possible to improve efficiency and optimise testing procedures.
   - Inheritance Complexity: Optimising the number of classes that inherit from each other to prevent accidentally breaking the code by modifying one class.

Results

When coming into this project there was a main and a secondary goal. Successfully creating and implementing GeoNotes into the existing Precision IQ (PIQ) android application was the main goal over this field session. Along the way we would be completing the secondary goal, how the splice API interfaced with the PIQ app for which it was created and the overall ease an extension can be created. As of now, all of the UI implementation is complete. GeoNotes can be created and added to the map and list. Pictures can be easily associated with a GeoNote as well. In addition the audio framework is in place for future expansion of our system. GeoNotes that are created are successfully placed into the database with a name, location, icon, and picture.

At this time, the GeoNote class and the database have been extensively tested using JUnit testing. These unit tests include testing the existence and accuracy of all data that comprises a GeoNote and successful creation of a GeoNote both from the source code and from the user interface. Unit tests are also used to verify the operations of the database in conjunction with the GeoNote class. Unit tests verify that: a GeoNote is placed in the database after creation; GeoNotes can be pulled from the database; and GeoNotes can be deleted from the database. The user interface has been tested by each member of the team individually by navigating through the app while taking notes to make sure all paths work, as well as to take note of anything that could be streamlined. At the moment all unit tests are passing and the UI is working as expected but further unit testing and UI testing will be performed on the image and audio functionality when it is completed.

Fortunately this project does not require an installation of important or otherwise unique software other than the provided. Adding and expanding upon this software should be fairly straightforward with few hiccups. In regards to the project we were assigned the UI can always be cleaned up but functionally the project is expected to have all of the features that were originally requested. While the GeoNotes extension works functionally there are an infinite number of extensions that can be added to the app using the Splice extension API. This could be a music extension, a video extension or something more pertinent to a farmer.

Lessons Learned

Utilizing and researching the API documentation is an important step that was overlooked when first starting the project leading to many issues that could have been avoided. GitHub and Android Studio do not mesh together well, leading to a significant amount of time spent on trying to get the two applications to work in tandem. In addition certain files are uniquely generated by Android Studio when the project is imported. Initially we all attempted to use the same generated files not knowing they had to be unique to everyone. Furthermore, the software was designed to work on an older android operating system requiring that we change certain aspects of the application we were using. This led to some of the documentation not being up to date with the current standard i.e newer features were not covered in the documentation. Finally, issues arose when attempting to communicate between fragments of the application. Unique methods had to be researched and employed so the GeoNotes extension could communicate within the Precision IQ application.

Future Work

In the future, the GeoNote extension has the potential to have many additions and modifications. First, the framework to add audio recordings to GeoNotes has been implemented. This leaves the opportunity for future developers to easily expand upon the existing framework. Next, the details page could be modified to allow users to edit the GeoNotes they have already created. The GeoNote extension was engineered so implementing this functionality would not require the modification of many files. The addition of view model and view model factory files allows software developers to easily implement any database operation without the risk of harming other aspects of the software. In addition, knowing that many of the newer applications are coded in Kotlin, the database files are coded in this language to provide a start towards switching this app to the new language in the future. Finally, the map shown in the GeoNote Extension Page can be modified to provide more functionality to the application.

The MapBox has plenty of opportunities to be expanded. The current implementation is lacking the unique icons used in the rest of the project. Due to time constraints a placeholder icon was used to represent GeoNotes, however using the icons from the rest of the project would allow a quicker way to identify points. The style and content of the information popups could be redesigned to better show information about a GeoNote, i.e. adding the pictures to the HTML document they load. The current drawing order for GeoNotes is above the tractor, which might cause confusion when placing multiple points near one another and driving past them.

Within this Android application there are numerous xml files that control the layouts of the different aspects of the GeoNote extension that were added onto PIQ. In the future, the layout of the components within the GeoNote extension can be optimised to improve the user experience even further. In addition, the GeoNotes customization could also be expanded on. Currently, the GeoNotes are stored the same way every time, providing the necessary functionality to organize the list could improve this. These organization queries could include: alphabetically, GeoNote type, creation date, hazard level, or even distance from the tractor.

Another large source of possible future work comes from the many possibilities that come from the Splice API. Through this API, an infinite number of community made extensions can be created. The GeoNote extension proves that it is possible to add a lot of functionality to Precision IQ in a relatively short amount of time. As a result additional extensions can also be added to the Precision IQ software.

Appendices:

This software has not been run on an actual tablet from Trimble, only the emulated counterpart. The emulated version does not perfectly represent it's real world counterpart so errors previously hidden may reveal themselves when subject to a real world test. Along with that the SDK version that we are using may not be compatible with the g12 tablets. Although this has not caused any issues when using the emulator.

The MapBox API uses an access token in order to access the service. We as a group have access to this token but for outsides using this software they will need their own key.