Yellowstone Ecological Research Group

Mobile Data Collection Application

Natalia Eigner, Grayson O'Leary

Montana State University

Spring 2022

Table of Contents

Introduction

The Yellowstone Ecological Research Center (YERC) aims to provide communities surrounding the Greater Yellowstone Region with comprehensive data on the health and changes of this ecosystem in order to empower data-driven decision making. YERC utilizes various forms of data collection including satellite imagery, weather history, environmental sensors and infield data collection. Currently, users manually transfer this data to YERC's online database, EPIIC. This method is not only inefficient for users, but also increases errors and creates a lack of standardization in the data. This cross-platform mobile application will replace manual collection and transfer, thus streamlining the data collection process. Users log in with their YERC credentials and the user preferences associated with their account will determine which data forms they have access to. Users are then able to access these data forms remotely and are able to upload them to the EPIIC platform when they are back in service. The data forms are populated directly from YERC's database and can thus be updated by administrators asynchronously without need to update the mobile application. This will also allow YERC to develop new data forms in the future as needed. In sum, this mobile application will streamline the data collection process of the Yellowstone Ecological Research Center resulting in more accurate and standardized data.

Qualifications

Natalia Eigner

970.301.8097
teigner30@gmail.com
https://www.linkedin.com/in/natalia-eigner-5123071ba/
https://teigner30.wixsite.com/my-site

Skills & Classes

Java, C, C#, Python, OpenCV, OpenGL, HTML, CSS, UML, Markdown (basic knowledge of Fortran, Go, Lisp)
Tableau, Excel, Adobe Bridge, Lightroom, Premiere Pro, Photoshop
Computer Vision, Data Structures and Algorithms, Software Engineering, Linear Algebra, CS Theory, Networks, Computational Biology

Experience

SensorLogic/ Marketing & Engineering Support Technician
AUGUST 2021 - PRESENT BOZEMAN, MT
This role includes support for SensorLogic's engineering team through testing products and creating documentation such as user guides. Further responsibilities include the creation and maintenance of marketing materials.

TechLink / Intern
JANUARY 2021 - AUGUST 2021 BOZEMAN, MT
Intern in TechLink's Software Engineering and Analysis laboratory which specializes in creating testing to analyze the security and overall capability of the Department of Defense's software programs.

University Studies Advising Office / Administrative Assistant
FEBRUARY 2019 - JANUARY 2021 MONTANA STATE UNIVERSITY
General organization and management of the office through communicating between students and advisors via Microsoft teams and other communication tools to manage student appointments. Also responsible for other administrative tasks and the creation of advising materials.

Gallatin Valley Botanical Farm / Production Manager
JUNE 2020 - AUGUST 2020, BOZEMAN, MT
Lead the processing and packaging of the Gallatin Valley Botanical Farm's produce for distribution to local restaurants, grocery stores, and individuals. This role required significant organization, efficiency, and attention to detail as well as the ability to lead the other farmhands throughout each stage of production.

Education

Montana State University / Computer Science, Photography, Honors College
AUGUST 2018 - PRESENT
GPA 3.7
Bachelor of science in computer science paired with a minor in photography.

Niwot High School / International Baccalaureate Program (IB)
AUGUST 2013 - MAY 2017
International Baccalaureate Program diploma recipient.

Community Engagement

Girls Who Code volunteer
Eagle Mount adaptive skiing volunteer with disabled children and adults
Leadership and Graphic Design with Montana State University Cru

Grayson O'Leary
114 Julia Martin Drive, Apt G
Bozeman, MT 59715
olearygrayson@gmail.com
209-597-2538

Objective: To work with YERC for my capstone project.

Education:

Montana State University. 201 Strand Union Building Bozeman, Mt. 59717—Aug. 2018-Present.

- Currently seeking Bachelor of Science in Computer Science, going on fourth year
- Relevant classes include: CSCI 127, 112, 232, 305, 322, 351, 356, 440, 460, and 466.

Work Experience:

MSU Catering Services in Bozeman from August of 2019 to March of 2020

- Worked as a dishwasher who would clean any pot, pan, or dish that needed to be cleaned. I was required to learn how to juggle and prioritize certain tasks on the fly.

MSU Alumni Foundation in Bozeman as a Fundraiser October of 2020 to May of 2021

- Work as a caller who seeks donations. In addition to being required to talk to potential prospects to build rapport and ask if they would like to donate.

MSU Residence Life in Bozeman as a Weekend Student Custodian from August of 2020 to May of 2021

- Clean my assigned dorm each weekend. In addition to being on call for two nights each month.

MSU Auxiliary Services in Bozeman as a Student Custodian from August of 2021 to Present.

- Clean MSU dining halls twice a week.

Awards:

- Received the International Baccalaureate Degree from Tracy High School
- Received a Seal of Biliteracy in French

*References available upon request.

Background

The Yellowstone Ecological Research Center (YERC) is committed to restoring and protecting the fragile ecosystem of the Greater Yellowstone Region. They believe that in order to do this, all community stakeholders must be provided with a comprehensive understanding of the health of these ecosystems through scientific data[1]. YERC utilizes a variety of sources to establish this holistic collection of ecological data including satellite imagery, weather history, environmental sensors and manual, infield data collection. They have recently created an platform, EPIIC, that allows for this data to be input into their database through an online dashboard. The infield data collection, however, is still done manually using a pen and paper and transferred onto the dashboard. This method is not only cumbersome for users, but also increases errors in their data. While YERC had a mobile application for data collection in the past, it was not robust enough to sustain upgrades and is no longer functional. The goal of this project, therefore, was to create a cross-platform mobile application that will streamline the data collection process. The application is robust enough to sustain device and library upgrades and can adapt as the needs and tools used by YERC change. This application also interfaces seamlessly with the EPIIC platform and allows users to both save forms to the device in the field and upload this data when back in service.

The project outlined in this document is specific to YERC and their processes. As a result, it is important to define the semantics used by YERC for their data collection process. A *field project* defines what kind of data an individual is collecting, whether it be a soil sample, water quality, or fish tag. Each field project has a *sample design* which defines what kinds of observations are being collected (i.e. temperature, pH, date, photos, etc.), where and how these observations are collected, and the field project's time frame. The *protocol* informs users when they are in the field exactly how to carry out the sample collection according to the sample design. The observations specified by the sample design are then recorded into a *data form.* The data forms differ for each field project and are defined in the EPIIC platform.

It is important to note that there is a similar tool to what we created called Survey123 from ArcGIS[2]. This tool, however, has some key differences from our solution. First, our application interacts directly and seamlessly with the EPIIC platform and their database without the extra steps needed to adapt Survey123 to YERC's specific needs. Our solution can also be utilized by users with little to no initial training or guidance, unlike Survey123. We have accomplished this through a simple interface and clear application flow. Survey123 is also limited to infield data collection, while our mobile

[1] https://www.yellowstoneresearch.org/
[2] https://survey123.arcgis.com/

application will provide YERC with the ability in the future to integrate other tools such as mammal tracking and environmental sensors if needed.

Work Schedule

To complete this project we utilized the Agile development life cycle. This iterative development life cycle allowed us to continually refine our mobile application so that the final product is a holistic solution to the problem presented by YERC. We began by developing the basic structure for each page of the application. This allowed us to test the flow of the application and refine the structure as needed. We then added the basic front-end functionality for each page, making sure that all potential use cases were considered and accounted for in the design. At the same time, one of our team members worked on the back-end functionality, linking the front end to pull data from YERC's database. Finally, the remaining time was used to add the rest of the requirements for the front-end and to test the application. We kept track of each milestone using GitLab.

Our initial schedule was as follows:

Milestone	Assigned To	Time Frame	Duration
Initial Exploration and Decisions	Both	07/21 - 01/22	
Front End			
Create application pages - basic structure and content	Tali	01/19/22 - 01/28/22	9 days
Establish overall architecture and application flow	Grayson	01/19/22 - 01/28/22	9 days
Test then integrate feedback	Both	02/01/22 - 02/08/22	7 days
Back End			
Storage and search for dataforms	Tali	02/09/22 - 02/23/22	14 days
Upload function	Tali	02/24/22 - 03/10/22	14 days
Link upload with EPIIC	Tali	03/11/22 - 03/25/22	14 days
Create JSON link between EPIIC and data form format	Grayson	02/09/22 - 02/23/22	14 days
Link login credentials from EPIIC to mobile app	Grayson	02/24/22 - 03/10/22	14 days

Create new account functionality	Grayson	03/11/22 - 03/25/22	14 days
Test then integrate feedback	Both	03/25/22 - 04/01/22	7 days
Final Steps			
Add functionality for other capstone protects if time allows	Both	04/01/22 - 04/15/22	14 days
Stress test application in the field	Both	04/01/22 - 04/15/22	14 days
Revise any errors	Both	04/15/22 - 04/22/22	7 days
Create documentation	Tali	04/22/22 - 04/27/22	5 days

Inevitably, certain milestones in this schedule took more time than we anticipated while others took less. As a result, our work schedule this semester looked like this:

Milestone	Assigned to	Time Frame
App screenflow	Tali	12/09/2021 - 1/25/2022
App handshake	Grayson	12/09/2021 - 1/25/2022
Create home page	Tali	1/26/2022 - 2/15/2022
Link login query with page	Grayson	1/26/2022 - 3/4/2022
Create consistent app design	Tali	2/2/2022 - 2/15/2022
Create sample JSONs for data forms	Grayson	2/9/2022 - 2/17/2022
Revise file structure	Tali	2/15/2022 - 2/15/2022
Create observation type selection that updates app state	Tali	2/15/2022 - 3/7/2022
User preferences and data form query builder	Grayson	3/4/2022 - 4/22/2022
Update data form page to populate from JSON	Tali	3/8/2022 - 4/8/2022
Enable dropdown widget	Tali	3/25/2022 - 3/25/2022
Link API calls with data form building	Grayson	4/8/2022 - 4/22/2022
Display saved data forms on home page	Tali	4/14/2022 - 4/19/2022
Add ability to save data forms locally	Tali	4/12/2022 - 4/14/2022
Add photo picker field	Tali	4/14/2022 - 4/28/2022

Proposal Statement

Functional

All of our primary functional requirements for this project were met. The user is able to log in to the application using their credentials which are then verified by the EPIIC platform. The application then queries the database and returns the user's available data form definitions. After they have logged in, the user can select a field project type and open the corresponding data form. The user can then record observations in this data form and save the data form to their device while they are in the field and potentially out of service. The data forms allow for text input, date selection, and image selection. The functionality to upload the data form to the EPIIC platform has been created, but as YERC has not yet created this API call, the upload cannot currently be completed. The user is also able to edit the data form after it has been saved. Potential further work for this project includes the ability to apply different filters for sorting the saved data forms and adding more specialized field types to the data forms.

Non-Functional Requirements

We have met the non-functional requirements of using JSON files to update the data forms for each field project type. This will allow YERC administrators to edit the data forms or add new ones without having to update the entire application. We have also met the non-functional requirement of having a clear and easy to navigate interface so that the application can be used out in the field without intensive instruction.

Performance Requirements

We have met the primary performance requirements of being able to save data forms to the device when users are in an area without service. There is also adequate storage allocated for saving the data forms, ensuring that no data is lost.

Interface Requirements

The primary interfaces for our application are communication between the user and the mobile application and between the mobile application and the database. We have met these requirements by allowing for the data forms to both be saved to the device and uploaded to the EPIIC database. As was stated previously, data forms cannot currently be uploaded to the database because this API call

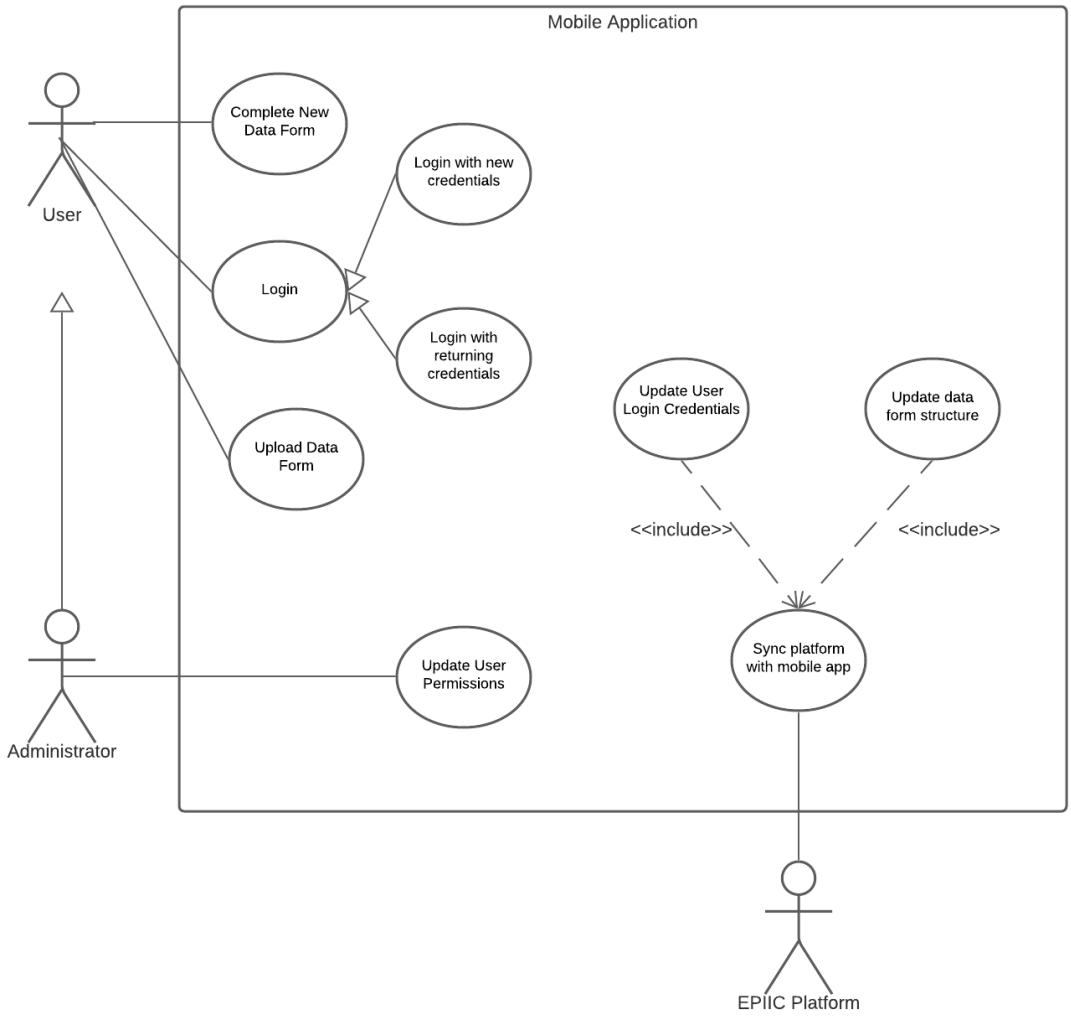
has not yet been created, but the functionality for this is present in the code. The design and flow of the interface is also easy to navigate and interact with so that minimal effort is needed from the user to use this application.

Development Standards and Tools Used

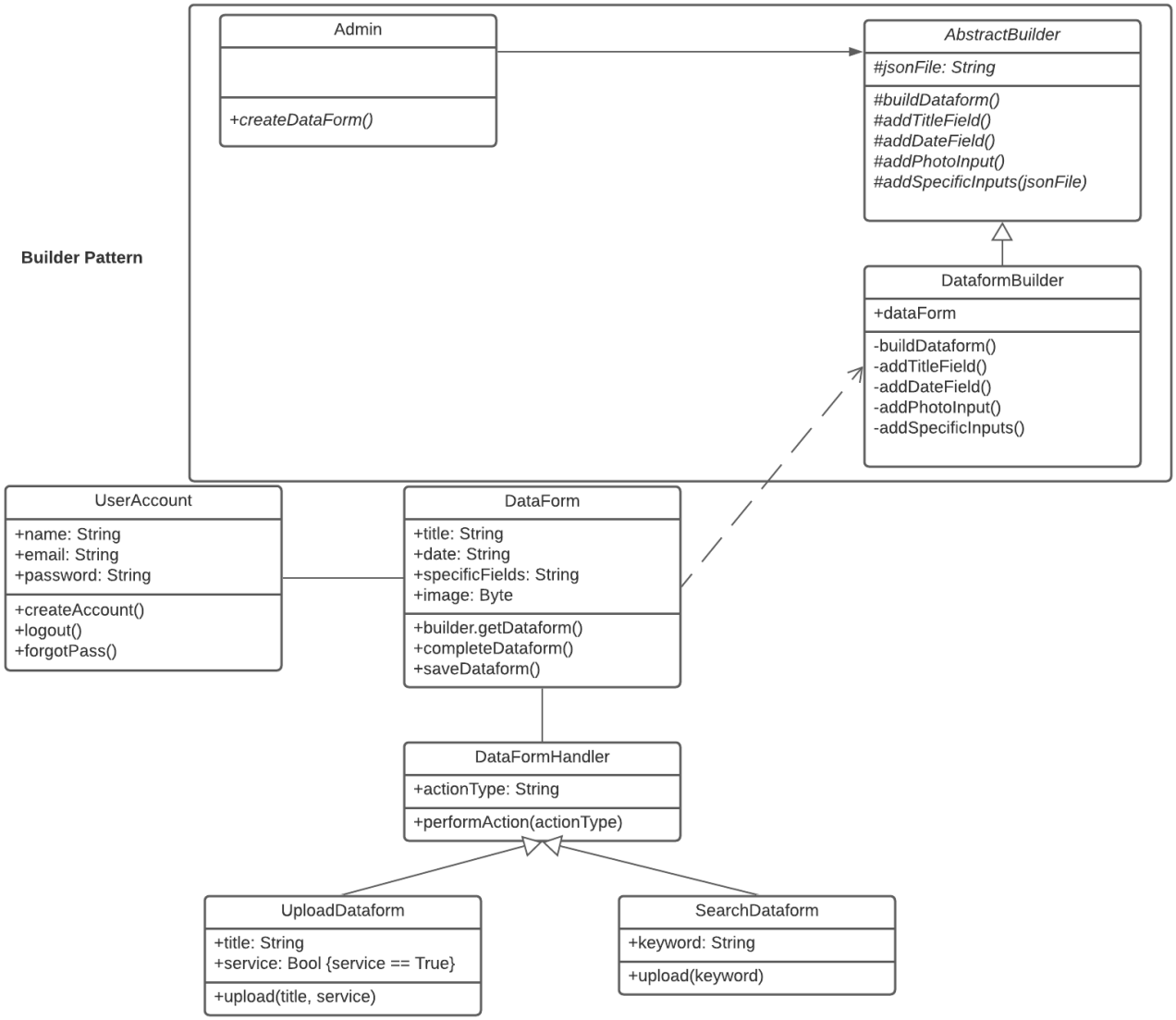
The mobile application was developed using Flutter, a UI development kit that utilizes the programming language Dart. Developers on Mac systems used IntelliJ as an IDE, XCode for simulating on iOS devices and Android studio for simulating on Android devices. Developers on Windows used Visual Studio Code. GitLab was used for version control and for tracking and resolving issues. Communication between the team and with our stakeholder took place over Slack. The application was tested on simulated devices as well as with the EPIIC platform for compatibility.

Methodology

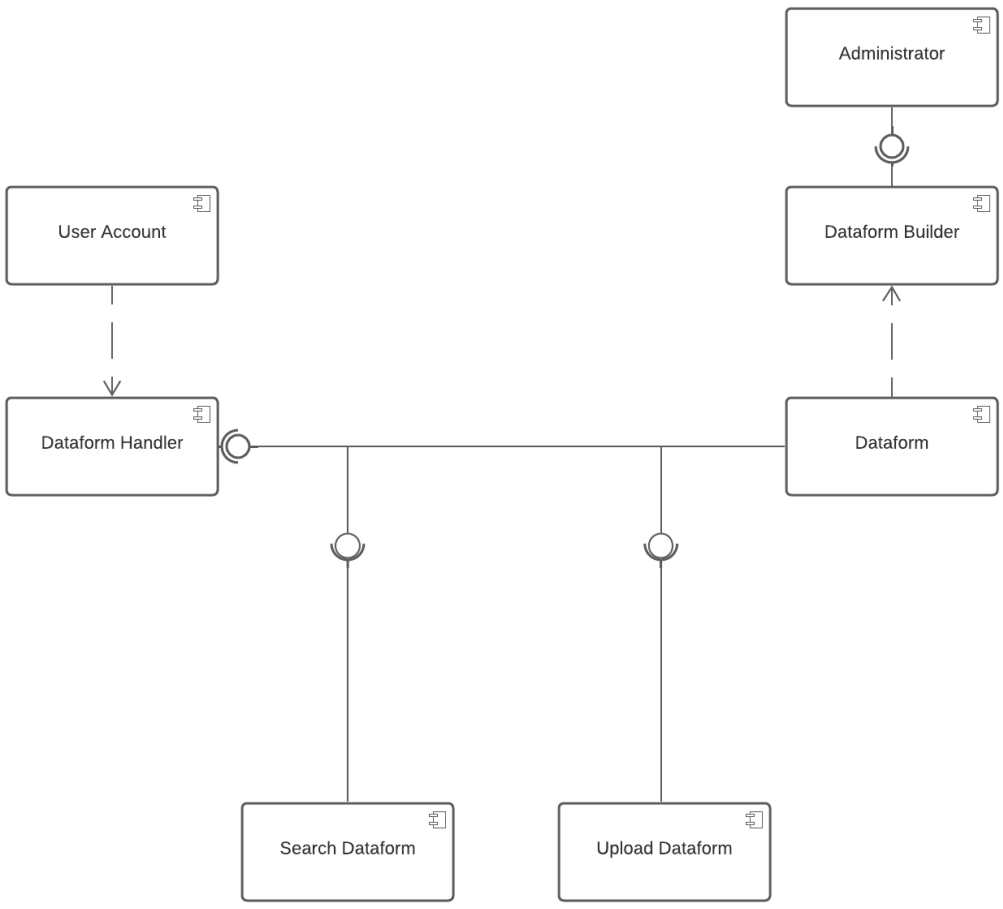
Use Case Diagram



Class Diagram



Component Diagram



As is evident in our class diagram, the builder pattern was implemented in our project design. We chose this pattern as our specifications were to communicate updates from the EPIIC platform to the data form structure on the mobile application. Thus, the application implements a data form builder that builds a data form to the specifications set via an input JSON file. The application then builds this data form according to these specifications. As a result, users will not have to manually update their applications every time an update is applied to EPIIC. Instead, the application updates its list of data forms and their structure when the user logs in while in service.

Our first course of action with this project was to decide what languages and software development kit to use. In our decision making process, we singled out two of the biggest and most feature complete mobile application development kits on the market: React Native and Flutter[3]. In

[3] https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021

making the decision between these two, we noted that they were very similar in the features they offer: cross-platform compatibility, hot reload, and an extensive package selection. Since they are very similar, our choice between the two came down to the difference between writing in JavaScript with React or Dart with Flutter. As Dart is a newer language and seems to be preferred over JavaScript for mobile applications we chose Flutter.

Another design choice we made was to integrate the data form types as a JSON file instead of hard coding them into the application. By using a JSON file YERC administrators will be able to edit the JSON files on the database and the mobile application will automatically receive these updates when the user logs in and will populate the data forms accordingly. As a result administrators will not have to update and redistribute the application every time a data form needs to be updated or added.

Appendix

Design Pattern

The Builder Pattern is implemented in json_to_form.dart and data_form_page.dart. A data form builder is created within json_to_form.dart. The type of data form and its required field types are encoded in a JSON and sent to the builder. The builder then creates this unique data form within the structure created in the data form page. For example, an aquatic insect data form will require a time field, two text input fields for number of kick samples and notes, and a photo selection field.

Code

main.dart

```
import …
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'YERC Mobile Application',
      theme: ThemeData(
        // Theme of application.
        primarySwatch: Colors.lightGreen,
      ),
      home: const LoginPage(title: 'YERC'),
    );
  }
}
```

data_form_page.dart

```
import …
class DataFormPage extends StatefulWidget {
  const DataFormPage({
    required this.storage,
    required this.Form,
    required this.typeState,
  });
  final FileStorage storage;
  final String Form;
  final String typeState;
  @override
  _DataFormPageState createState() => _DataFormPageState();
}
class _DataFormPageState extends State<DataFormPage> {
  dynamic response;
  File? image;

  Icon fab = const Icon(Icons.check_box_outline_blank);
  int fabIconNumber = 0;
  generateFileName() {
    String dataFormId = widget.typeState.replaceAll(' ', '-');
    DateTime curDate = DateTime.now();
    String formattedDate = DateFormat('yyyy-MM-dd–kk:mm').format(curDate).toString();
```

```
  String filePath = dataFormId + formattedDate;
  return filePath;
}
@override
Widget build(BuildContext context) {
  Map formMap = json.decode(widget.Form);
  return Scaffold(
    appBar: AppBar(
      title: Text('DataForm Page', style: TextStyle(fontSize: 16, color: Colors.grey[800])),
      backgroundColor: Colors.white,
      automaticallyImplyLeading: false,
    ),
    body: Row(
      children: <Widget>[
        Expanded(
          child: JsonForm(
              storage: FileStorage(),
              formMap: formMap,
              typeState: widget.typeState,
              onChanged: (dynamic response){
                this.response = response;
              }
          ),
        ),
      ],
    ),
    floatingActionButton: Wrap(
      children: <Widget>[
        Container(
          margin:EdgeInsets.all(5),
          height: 40,
          width: 80,
          child: FloatingActionButton(
            heroTag: "cancelBtn",
            onPressed: () {
              Navigator.pop(context);
            },
            child: Text('Cancel', style: TextStyle(fontSize: 12)),
            backgroundColor: Colors.red[800],
            foregroundColor: Colors.white,
            shape: RoundedRectangleBorder(borderRadius: BorderRadius.all(Radius.circular(3.0))),
          ),
        ),
        Container(
          margin:EdgeInsets.all(5),
          height: 40,
          width: 80,
          child: FloatingActionButton(
            heroTag: "saveBtn",
            onPressed: () {
              for(var count = 0; count < formMap['dataFormsByIds'].length; count++){
                if(formMap['dataFormsByIds'][count]['name'] == widget.typeState){
                  if(formMap['dataFormsByIds'][count]['filePath'] == ''){
                    String fileName = generateFileName();
                    formMap['dataFormsByIds'][count]['filePath'] = fileName;
                    widget.storage.writeDataform(response, fileName);
                    Navigator.pop(context);
                  }
                  else{
                    String fileName = formMap['dataFormsByIds'][count]['filePath'];
```

Dataform builder is called in the dataform page

```
                widget.storage.writeDataform(response, fileName);
                Navigator.pop(context);
              }
            }
          }
        },
        child: Text('Save', style: TextStyle(fontSize: 12)),
        backgroundColor: Colors.lightGreen[600],
        foregroundColor: Colors.white,
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.all(Radius.circular(3.0))),
      ),
    ),
  ],
 ),
 );
 }
}
```

json_to_form.dart

```
import …
class JsonForm extends StatefulWidget {
 const JsonForm({
  required this.formMap,
  required this.onChanged,
  required this.storage,
  required this.typeState,
 });
 final Map formMap;
 final ValueChanged<dynamic> onChanged;
 final FileStorage storage;
 final String typeState;
 @override
 _JsonFormState createState() => _JsonFormState(formMap);
}
class _JsonFormState extends State<JsonForm> {
 final dynamic formGeneral;
 // Initialize 2d array for pairs (fieldname, value)
 dynamic response;
 _JsonFormState(this.formGeneral);
 @override
 Widget build(BuildContext context) {
  return Padding(
   padding: const EdgeInsets.symmetric(horizontal: 5.0),
   child: jsonToForm(),
  );
 }
 Widget jsonToForm(){
  final children = <Widget> [];
  for (var count1 = 0; count1 < formGeneral['dataFormsByIds'].length; count1++){
   if (formGeneral['dataFormsByIds'][count1]['name'] == widget.typeState){
    children.add(
     Padding(
      padding: EdgeInsets.only(top: 20, bottom: 20),
       child: Text(formGeneral['dataFormsByIds'][count1]['name'] + " Observation", style: TextStyle(fontSize: 26)))
     );
    for (var count = 0; count < formGeneral['dataFormsByIds'][count1]['attributes'].length; count++) {
     response = List.generate(count, (i) => " ", growable: true);
     Map item = formGeneral['dataFormsByIds'][count1]['attributes'][count];
     if(item['widgetType'] == 'input' || item['widgetType'] == 'input') {
      children.add(ListTile(
```

```
            title: SimpleText(
              name: item['name'],
              labelText: item['labelText'],
              hintText: item['hintText'],
              required: item['required'],
              inputValue: item['inputValue'],
              widgetType: item['widgetType'],
              response: response,
              position: count,
              pos1: count1,
              onChange: onChange,
            ),
          ));
        }
        if(item['widgetType'] == 'Date Picker' || item['widgetType'] == 'date picker') {
          children.add(ListTile(
            title: DateField(
              labelText: item['labelText'],
              inputValue: item['inputValue'],
              position: count,
              pos1: count1,
              onChange: onChange,
            ),
          ));
        }
        if(item['widgetType'] == 'attachment' || item['widgetType'] == 'attachment') {
          children.add(ListTile(
            title: PhotoField(
              labelText: item['labelText'],
              inputValue: item['inputValue'],
              position: count,
              pos1: count1,
              onChange: onChange,
            ),
          ));
        }
      }
    }
    return ListView(
      children: children,
    );
  }
  void _handleChanged() {
    widget.onChanged(formGeneral);
  }
  void onChange(int position, dynamic value, int pos1) {
    setState(() {
      formGeneral['dataFormsByIds'][pos1]['attributes'][position]['inputValue'] = value;
      _handleChanged();
    });
  }
}
```

Each dataform is built with unique specifications from JSON files pulled from EPIIC database

entry_fields.dart

```
import 'package:flutter/material.dart';
import 'data_form_page.dart';
import 'file_storage.dart';
class EntryField extends StatelessWidget {
  final String name;
```

```
  final String date;
  final FileStorage storage;
  final dynamic fileContents;
  final String typeState;
  EntryField({
    required this.name,
    required this.date,
    required this.storage,
    required this.fileContents,
    required this.typeState,
  });
  @override
  Widget build(BuildContext context) {
    return Container(
      margin: const EdgeInsets.only(top: 5.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          SizedBox(height: 12),
          Column(
            children: <Widget> [
              Text(name, style: TextStyle(fontWeight: FontWeight.bold)),
              Row(
                crossAxisAlignment: CrossAxisAlignment.center,
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                children: <Widget> [
                  Container(
                    child: Text(date),
                  ),
                  Container(
                    child: IconButton(
                      icon: Icon(Icons.create_rounded, color: Colors.grey[800]),
                      onPressed: () {
                        Navigator.of(context).push(MaterialPageRoute(builder: (context) => DataFormPage(storage: FileStorage(),Form:
fileContents, typeState: typeState,)));
                      },
                    ),
                  ),
                  Container(
                    child: IconButton(
                      icon: Icon(Icons.cloud_upload, color: Colors.grey[800]),
                      onPressed: () {
                      },
                    ),
                  ),
                ],
              ),
            ],
          ),
          SizedBox(height: 12),
          Container(height: 2, color: Colors.lightGreen[600]),
        ],
      ),
    );
  }
}
```

file_to_entry.dart

```
import …
```

```
class FileEntry extends StatefulWidget {
  const FileEntry({
    required this.fileNames,
    required this.storage,
    required this.typeState,
  });
  final List fileNames;
  final FileStorage storage;
  final String typeState;

  @override
  _FileEntryState createState() => _FileEntryState();
}

class _FileEntryState extends State<FileEntry> {
  String name = '';
  String date = '';
  int loc = 0;
  _FileEntryState();

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 5.0),
        child: fileToEntry(),
    );
  }

  Widget fileToEntry(){
    final children = <Widget> [];
      for (var count = 0; count < widget.fileNames.length; count++) {
        final extension = p.extension(widget.fileNames[count]);
        if(extension == '.json') {
          File(widget.fileNames[count]).readAsString().then((String contents) {
            dynamic jsonContents = json.decode(contents);
              name = jsonContents['dataFormsByIds'][loc]['name'];
              //name = jsonContents['dataFormsByIds']['filePath'];
              //name = name.replaceAll('-',' ');
              for (var count = 0; count < jsonContents['dataFormsByIds'][loc]['attributes'].length; count++) {
                Map item = jsonContents['dataFormsByIds'][loc]['attributes'][count];
                if (item['widgetType'] == 'Date Picker' || item['widgetType'] == 'date picker') {
                  date = item['inputValue'];
                  children.add(ListTile(
                    title: EntryField(
                      date: date,
                      name: name,
                      storage: widget.storage,
                      fileContents: contents,
                      typeState: widget.typeState,
                    ),
                  ));
                }
              }
          });
        }
      }
      return ListView(
        children: children,
      );
  }
```

```
}
```

login_page.dart

```
import …
import 'package:http/http.dart' as http;
import 'home_page.dart';
import 'file_storage.dart';

class loginInfo {
  final String user;
  final String pass;
  loginInfo(this.user, this.pass);
  Map<String, dynamic> toJson() => {
    'username': user,
    'password': pass,
  };
}
Future<bool> signIn(http.Client client, String loginInfo) async{
  final headers = {"Content-type": "application/json"};
  final response = await client.post(Uri.parse("https://epiic-api-dev.azurewebsites.net/api/auth/signin"),  headers: headers,
body: loginInfo);
  if (response.statusCode == 200){
    Map<String, dynamic> body = jsonDecode(response.body);
    var $YOUR_PERSONAL_ACCESS_TOKEN = body["access_token"];
    final _httpLink = HttpLink(
      'https://epiic-api-test.azurewebsites.net/graphql',
    );
    final _authLink = AuthLink(
      getToken: () async => ('Bearer ' + $YOUR_PERSONAL_ACCESS_TOKEN),
    );
    Link _link = _authLink.concat(_httpLink);
    final GraphQLClient client = GraphQLClient(
      cache: GraphQLCache(),
      link: _link,
    );
    const String queryUserPrefs = r'''
      {
        dataFormsByIds(ids: "aquatic-insects") {
        id
        name
        attributes {
          name
          labelText
          hintText
          required
          dataType
          unit
          widgetType
          }
        }
      }
      ''';
    final QueryOptions options = QueryOptions(
        document: gql(queryUserPrefs)
    );
    final QueryResult result = await client.query(options);
    if (result.hasException) {
      print(result.exception.toString());
    } else {
      print(result);
```

```
    }
    return true;
  } else {
    return false;
  }
}
class LoginPage extends StatefulWidget {
  const LoginPage({Key? key, required this.title}) : super(key: key);
  final String title;
  @override
  State<LoginPage> createState() => _LoginPageState();
}
class _LoginPageState extends State<LoginPage> {
  @override
  Widget build(BuildContext context) {
    String user = "";
    String pass = "";
    return Scaffold(
      body: SingleChildScrollView(
        child: Container(
          color: Colors.grey[200],
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          child: Column(
            children: <Widget>[
              Padding(
                padding: const EdgeInsets.only(bottom: 10.0),
                child: Center(
                  child: Container(
                    width: MediaQuery.of(context).size.width,
                    //height: 150,
                    child: Stack(
                      children: <Widget>[
                        Container(
                          child: Image.asset('assets/images/yellowstone.jpg'),
                        ),
                        Container(
                          margin: const EdgeInsets.all(30.0),
                          child: Row(
                            children: <Widget>[
                              Container(
                                child: Image.asset('assets/images/yerc_logo.png', height: 30, width: 60),
                              ),
                              Container(
                                margin: const EdgeInsets.all(10),
                                child: Text("YERC", style: TextStyle(fontWeight: FontWeight.bold, fontSize: 40, color: Colors.white)),
                              ),
                            ],
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
              ),
              Padding(
                padding: EdgeInsets.symmetric(horizontal: 15),
                child: TextField(
                  onChanged: (text){
                    user = text;
```

```
    },
    decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: 'Email',
        hintText: 'ie: abc@example.com'
    ),
  ),
),
Padding(
  padding: EdgeInsets.only(left: 15, right: 15, top: 15),
  child: TextField(
    onChanged: (text){
      pass = text;
    },
    decoration: InputDecoration(
      border: OutlineInputBorder(),
      labelText: 'Password',
      hintText: 'Password',
    ),
    obscureText: true,
  ),
),
Padding(
  padding: EdgeInsets.all(20),
  child: Container(
    height: 40,
    width: 150,
    decoration: BoxDecoration(
        color: Colors.grey[800], borderRadius: BorderRadius.circular(10)
    ),
    child: FlatButton(
      onPressed: () async{

        loginInfo signin = loginInfo(user, pass);
        var success = await signIn(http.Client(), jsonEncode(signin.toJson()));
                if (success){
                  print("go");
                  Navigator.of(context).push(MaterialPageRoute(builder: (context) => HomePage(storage:FileStorage(),)));
                } else {
                    showDialog(
                        context: context,
                        builder: (context) {
                            return Dialog(
                                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(40)),
                                elevation: 16,
                                child: Column(
                                    children: <Widget>[
                                        Padding(
                                            padding: EdgeInsets.all(14),
                                            child: Text(
                                                "Error",
                                                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 16, color: Colors.red),
                                            ),
                                        ),
                                        Padding(
                                            padding: EdgeInsets.all(14),
                                            child: Text(
                                                "Incorrect username or password. Please try again or click Forgot Password.",
                                                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 16, color: Colors.red),
                                            ),
```

```
                                        ),
                                    ],
                                ),
                            );
                        }
                    );
                }
            },
            child: Text(
              'LOGIN',
              style: TextStyle(color: Colors.white, fontSize: 18),
            ),
          ),
        ),
        ),
      ],
    ),
  ),
 );
 }
}
```

home_page.dart

```
import …
String json = "";
class JsonReturn {
  String timestamp = "";
  String collectionMethod = "";
  String collectionTime = "";
  String dataset = "";
  String filename = "";
  String mobId = "";
  String name = "";
  String value = "";
  JsonReturn(Map<String, dynamic> json){
    timestamp = json['Timestamp'];
    collectionMethod = json['collection_method'];
    collectionTime = json['collection_time'];
    dataset = json['dataset'];
    filename = json['filename'];
    mobId = json['mob_id'];
    name = json['name'];
    value = json['value'];
  }
  Map<String, dynamic> toJson() => {
        'Timestamp': timestamp,
        'collection_method': collectionMethod,
        'collection_time': collectionTime,
        'dataset': dataset,
        'filename': filename,
        'mob_id': mobId,
        'name': name,
        'value': value
      };
}
class HomePage extends StatefulWidget {
  HomePage({
    required this.storage,
    required this.inJson,
```

```
  });
  final FileStorage storage;
  final String inJson;
  @override
  _HomePageState createState() => _HomePageState();
}
class _HomePageState extends State<HomePage> {
  bool box = false;
  var userList = [];
  String curState = '';
  String readFile = '';
  String name = '';
  dynamic attributes = '';
  String date = '';
  String fileContents = '';
  List fileNames = [];
  @override
  void initState() {
    userList.add(Modal(name: 'Aquatic Insects', isSelected: false));
    userList.add(Modal(name: 'Water Quality', isSelected: false));
    super.initState();
    getFiles();
  }
  getFiles() async {
    final directory = await getApplicationDocumentsDirectory();
    final path = directory.path;
    final dir = Directory(path);
    final List<FileSystemEntity> entities = await dir.list().toList();
    final Iterable<File> files = entities.whereType<File>();
    String strFiles = files.toString().replaceAll('File: ', '');
    strFiles = strFiles.replaceAll("')", '');
    strFiles = strFiles.replaceAll("('", '');
    fileNames = strFiles.split("', '");
    return fileNames;
  }
  @override
  Widget build(BuildContext context) {
    getFiles();
    print('File names in home page');
    print(fileNames);
    return Scaffold(
      appBar: AppBar(
        title: Text("Field Project: " + curState, style: TextStyle(fontSize: 16, color: Colors.grey[800])),
        backgroundColor: Colors.white,
        actions: <Widget>[
          Container(
            padding: EdgeInsets.all(0),
            width: 60,
            child: FlatButton(
              onPressed: (){
                showDialog(
                  context: context,
                  builder: (context) {
                    return Dialog(
                      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(40)),
                      elevation: 16,
                      child: Stack(
                        alignment: Alignment.center,
                        children: <Widget>[
                          Container(
```

```
                    width: MediaQuery.of(context).size.width,
                    height: 200,
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(15),
                    ),
                    padding: EdgeInsets.all(20),
                    child: const Text("Select a Field Project Type",
                        style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                        textAlign: TextAlign.center
                    ),
                  ),
                  ListView.builder(
                    shrinkWrap: true,
                    itemCount: userList.length,
                    itemBuilder: (context, index){
                      return Container(
                        padding: EdgeInsets.all(10),
                        height: 50,
                        child: Row(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          mainAxisAlignment: MainAxisAlignment.spaceBetween,
                          children: [
                            IconButton(
                              icon: _iconControl( userList[index].isSelected),
                              onPressed: () {
                                setState(() {
                                  userList.forEach((element) {
                                    element.isSelected = false;
                                  });
                                  userList[index].isSelected = true;
                                  curState = userList[index].name;
                                  (context as Element).reassemble();
                                });
                              },
                            ),
                            Text(
                                userList[index].name
                            ),
                          ], //Widget
                        ), //Row
                      );
                    },
                  ),
                ],
              ),
            );
          },
        ); //Show Dialog
      },
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(3)),
      child: Icon(Icons.create_rounded, color: Colors.lightGreen[600], size: 30),
    ),
  ),
 ],
),
body: Column(
  children: <Widget>[
    const Padding(
      padding: EdgeInsets.all(20),
    ),
```

```
        const Text("Saved Observations", style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold)),
        Expanded(
          child: FileEntry(
            storage: FileStorage(),
            fileNames: fileNames,
            typeState: curState,
          ),
        ),
      ],
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        if(curState == ''){
          showDialog(
            context: context,
            builder: (context) {
              return Dialog(
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(40)),
                elevation: 16,
                child: Stack(
                  alignment: Alignment.center,
                  children: <Widget>[
                    Container(
                      width: MediaQuery.of(context).size.width,
                      height: 100,
                      decoration: BoxDecoration(
                          borderRadius: BorderRadius.circular(15),
                      ),
                      padding: EdgeInsets.all(20),
                      child: const Text("Please select an field project type",
                          style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                          textAlign: TextAlign.center
                      ),
                    ),
                  ],
                ),
              ); //Return Dialog
            },
          );
        }
        else{
          Navigator.of(context).push(MaterialPageRoute(builder: (context) => DataFormPage(storage: FileStorage(), Form:
inJson, typeState: curState,)));
        }
      },
      child: const Icon(Icons.add, color: Colors.white),
      backgroundColor: Colors.lightGreen[600],
    ),
  );
 }

 _iconControl(bool box) {
   if (box == false) {
     return Icon(Icons.check_box_outline_blank_rounded);
   } else {
     return Icon(
       Icons.check_box_rounded,
       color: Colors.lightGreen[600],
     );
   }
```

```
  }
}

class Modal {
  String name;
  bool isSelected;
  Modal({required this.name, this.isSelected = false});
}
```

References

1. Yellowstone Ecological Research Center (YERC) - https://www.yellowstoneresearch.org/
2. ArcGIS Survey123 - https://www.esri.com/en-us/arcgis/products/arcgis-survey123/overview
3. EPIIC - https://www.yellowstoneresearch.org/epiiccenter
4. Flutter vs React - https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021
5. JSON - https://docs.flutter.dev/development/data-and-backend/json