Appendix A

Interviewer: All right so my first question is uh how long have you been teaching PE?

Interviewee: uh I qualified in 2009 and worked in a comprehensive school which means it has both boys and girls of all abilities in England in Kent which is a county in the southeast for nine years and then I moved here in about 2018 so this is my third year of the is so about 12 years?

Interviewer: Yeah

Interviewee: If that adds up?

Interviewer: Yeah, okay so um what do you use to do in the past to help your players?

Interviewee: Okay so there's a few computer programs and apps that we use so this thing's called uh coach's eye and other video recording software whereby so if you if you take for example a golf swing you can get your student swinging the ball with the camera watching from behind which has a which can take sort of like 10 000 frames per second pictures so then you can really minutely analyze different parts of the body and you can talk about the biomechanics of the swing how to increase swing speed and you can put that next to the elite model so it's a lot of here's the amateur which is the one you're working with and then the elite model how can you make it more like the elite model and other things we use are more data analysis software for teams so for example a cricket team which is a bit like some of the sports we play here like people whereby you can look at the data from matches to see where strengths and weaknesses are so if you look at like a player's batting wagon wheel which is where the lines go on the map of the batter to show where they hit the ball and what type of shots they hit so for example if you've got a batter that has had 10 innings and he's been caught at let's say mid on which for a right-handed batter is there we've been caught eight times in that position that sort of tells you as a coach well there must be something wrong with their technique that they're always chipping the ball up to the field or there for the surface event to get out so that's where you can use that type of software

Interviewer: Um so why do you find analyzing players to be a problem?

Interviewee: Um well it's it's an issue with some activities so things like t-ball, there aren't really- because t-balls are a kid's game that they don't have in the UK they don't they play softball if they're gonna play anything so it's a different game it has different um skills that are needed you need to collect different data points that you couldn't with other um analysis software i suppose so that's a difficulty for for sports that um aren't already featured on apps and another thing is it sometimes it can be time consuming for example if you're just one coach um and you've got to input data and coach and teach at the same time that can be difficult so something that's maybe a little bit more child friendly would be would be a good thing to have because then the kids can coach each other so let's say you've got a class of 24 and your team has 24 players you can get some people working on that aspect and also it's improving their skills as well to use computers and things so that could be a something to look into to make it more child-friendly. Also if possible they can view the my feedback and statistics with their own tablets or phones.

Interviewer: Oh yeah okay um so why do you think of this solution specifically?

Interviewee: uh just to try and improve performance really um and also if it is something the students can use is to improve their soft skills of using um ICT in class

Interviewer: Um to discuss more about like since uh I think we are going down the path of t-ball now?

Interviewee: yeah yeah

Interviewer: um can you uh give me your expertise on it because I don't think I know a lot about the game.

Interviewee: Okay so t-ball is a striking field in games you have one team that is in and they're they're trying to score runs that's the batting team and you have one team that's fielding it's very similar to baseball softball so the batter needs to hit the ball within a certain area of the field okay they then need to advance around the bases touching them in turns they've got to go to first place then second place then third base always at home base if you hit it really far maybe you could score a double so you go from first place to second base or you could even go first second third or if you hit it really far you can go around all of the bases at once and score at home run um the fielding team obviously needs to try and stop the ball and stop the uh opposing team scoring runs they need to spread themselves out they need to be able to stop the ball effectively catch the ball if it's in the air throw the ball accurately and with good distance but also have a knowledge of the tactics and the rules so in this child's game sometimes a ball hit in still inside the diamonds so not very far here the batter could actually get to like second or third base because the kids make mistakes they throw it to the wrong base or it takes them too long to stop it and pick it up um i mean in a nutshell that's about it there are a lot more complicated rules to it but that's okay you know that's a very basic level right you've got to hit it you're going to run around the bases the fielders have to stop it and try and get them out let's see

Interviewer: Okay wait so the t-ball is it like um where it has like the pole-

Interviewee: Yeah

Interviewer: The ball on top?

Interviewee: Yeah okay if you're free after school today you can come and watch our Fobisia team play so you can have a bit more of an idea or uh Tuesday morning Thursday morning we have i have classes if you want to come and watch her yeah a few games since you have a bit more of an idea or you could watch some baseball on Youtube because it's quite similar to baseball instead you don't pitch there's no throwing of the ball in the batter just hit it off the team.

Interviewer: I see okay. So um I mean I think yeah I think I asked you in this the questionnaire before but um can you uh talk to me through about like what data I need to record?

Interviewee: Right so there's a lot that you can do right even if you if you just think of batting

Interviewer: Yeah

Interviewee: So if you just think of just the one aspect of a player's performance so a batter will batter a certain number in the order so the order is who that's at what position okay so it goes one through to ten in t-ball so your first batter is normally a slightly weaker batter okay so what tends to happen is in some teams you'll have people who are particularly good at batting but then maybe their fielding isn't as good okay well I'll elaborate on that a little bit so let's say your first batter their main aim is to hit the ball well to get to first base okay your second batter their job is to hit the ball well enough so the base runner on first can get to second and they can get to first the third batter needs to be slightly better again to be able to hit the ball a bit further so that every player can get to the next base so then you've got a base runner on first place second base third base so then when the fourth batter comes up which needs to be one of your

big hitters you can hit the ball really far so all of them can get a run or two of them can get a run and then advance the next two on fifth similar to one I need to hit the ball get to first place so basically a tactic is you're trying to fill the bases up and then get your fourth and eighth batter to empty them right um so if you have your first batter absolutely blast the ball miles and run all the way around it's good you score a point but it would be better if you had some of your weaker players positioned on the bases so they can score as well so as a batter you can judge things like all the data points can be direction of hits so do they hit it left field right field and maybe something like a map of the field let me show you quickly a diagram. So something like this. Very simple. So the batter will be here and maybe you have something in the app for batting you can tap the screen on like an iPad so the ball where the ball bounces so it bounced there and then the next time about that's there that's here that's here and then you have a collection of data points see so you could draw. Do you play golf?

Interviewer: No, I don't

Interviewee: Okay they do this at the screen golf where where you hit your driver let's see hit it here here and in here it'll actually draw like a circle of best fit as to where it predicts your next shot is going to go so if you've got a batter therefore that has a lot in here that's not particularly good because it's likely that the picture is going to catch the ball through it so first, they'll be out first so just something that's easy wherever you could go right here here here here okay or another way if you can't code that. Because that's easy right even a kid can a kid could do that and they could put it, so that's like direction here you can also have a a column or a box about how many points they've scored so how many singles they've hit so if they run to first base you have another single double triple home run that's another data point you could have out so how did you get out were they tagged out? Were they forced out? Were they caught out? Did they strike out? So they should hit the ball three times against the team? Um I mean that's now already five four or five data points you're getting for a batter um caught out - where you are caught out - things like you could have unforced errors as a as an area for improvement. You have things like knowledge of the rules so kid… An advanced rule in t-ball is if you're on a base and fielder catches the ball you've got to run back and touch your base so if they ran on it shows that they don't know the rules. So maybe you can have some a qualitative data pointer in there so that the coach or the student could type, maybe? It's a comments area for things that you can't generate quantitative data on. Make sense? Um I don't know if there'd be a way to measure their speed on the app that might be a little bit more difficult how fast they are from home to first to first to second? Um…

Interviewer: Uh… I mean… You can use a timer instead?

Interviewee: Yeah? So, as they hit the ball you press the button and as they reach the base you press the button?

Interviewer: And like there could be a pre-programmed like distance

Interviewee: Yeah. And speed equals distance over time easy and then so that's just for batting there's probably even a few more that you could put in there as well if you if you thought about it but for for our needs that's probably that's probably about it I would say? We don't need to know a great deal more than that. Uh… Fielding… So you can have fielding by position: what position they field uh… How many times the ball came to them. how many times they stopped it effectively? Did they use a short barrier? Did they use a long barrier? How many catches have they made? How many drops did they make? How many times did they throw into a base and how many catches were made on a base? So, there's lots of fielding as well and then you could build up a big picture as to the um player's performance. Does this make sense to you even though it's a sport that you're not familiar with?

Interviewer: Yeah, yeah it makes sense to me.

Interviewee: Okay, so do you have any other questions?

Interviewer: Uhhhh… no.

Interviewee: Okay how- Are you making an app?

Interviewer: Yeah, an app

Interviewee: Okay…how come how complex does that happen to be?

Interviewer: Uhm… I think this level complexity is okay as it's like mostly data inputted and then we write down calculations and it's just it'll put a graph so it's not I wouldn't say that this this is the right level complexity

Interviewee: Okay so would it be too would it make it too difficult to do it both for batting and fielding or would you maybe be better off doing keeping it succinct just for batting maybe that might be something you want to ask and see what he thinks because otherwise it might get a little bit convoluted and quite difficult for you to do um so for example if the teacher only has one app one iPad and you're having to note down who's batting where they're hitting it who's fielding it how are they feeling it that's quite a lot you have to input yeah in one go so maybe you could just stick to batting perhaps that's just an idea for you to think about and something you could write about in like your I mean I'd imagine like a sort of science experiment like a introduction whatever or thoughts or there must be a paragraph you have to write somewhere about how you came to the conclusion of how to make the app yeah um you could talk about well originally we were thinking about making it through the whole batting and fielding but this is why it's a good thing this is why it's not so good for a primary PE teacher um so batting would be would be quite a good one I think to do but also where it input the data points and it gives you the the data comes back to you is there a way you could link? I can help you with it what that data means to a sentence. So for example if you keep hitting the ball here caught by the catcher or fielded by the catcher out at first you put the data in 10 data points forever and it will come out and saying "Your shots tend to go short inside the diamonds towards the picture this has resulted in you being out at first base 50 percent of the time". You see what I mean because then the kid will get feedback on that and I wouldn't have to write anything the app would do it for me and then maybe if that is if you get this comment you get to improve you need to try hitting the ball harder to this area of the field and maybe even here is a link to like a coaching video that'd be good right.

Interviewer: For the program I think it'd be better that we keep it succinct for now and focus on batting as that itself may be complex enough. Um also for like I think to calculate the percentage that they are caught out you need to put in like the points that they are like caught out so like maybe if they hit short but they didn't get caught out that that would be yellow they got caught out

Interviewee: Yeah well that's that's exactly what they do in cricket so in cricket… I don't even know what- it would be cool. colored dots. See there are some apps already on the market um this is probably from there where ah here so this is for a bowler okay so this is aren't you from a little cricket

Interviewer: Uh yeah, I think I played it before

Interviewee: All the bounces of the ball. Is the ball it bounces it comes towards the stumps so this will be uh ah so look so you've got select in sri lanka this is the batsman this is the bowlers data so every red dot the batsman didn't score every orange dot or peach colored dot the batter has scored runs by hitting the ball this side can you see because they're more towards the body it's easier for him to hit it that way

towards his legs any dark yellow was a four so he's hit it a long way can you see if that ball bounces and pitches here the batsman knows that's not going to hit the stun so he can have a free whack at that one so that's a four offside light blue he scored runs off these ones dark blue scored runs off these ones and then the other one might be wickets so how many times you got the player out so that's for bowling right so something similar you could do for a batting performance for t-ball um and have colors for base hit first base base hit second base basic third base base hit home run out caught out um at first out of seconds or something. Uh do you want to maybe go away and have a have a play around some ideas like on a I mean I would put it on like paint or draw what I'd wanted to look like you know like a planning stage so you're doing like a planning stage before you code it

Interviewer: Um yeah uh it's only next week that we start that

Interviewee: Okay okay, so if you start designing then we'll have another little catch up and we'll see what it looks like and it changes yeah? Happy? So you've got some things to be going on with now

Interviewer: Yeah

Interviewee: Okay, okay cheers then ! Thanks. Just email me when you want to come in next. Let me know all right?

Interviewer: Yeah. Thank you.

Interviewee: No problem

Appendix B

Why would you want this project to be completed?

1 response

To make it easy to see players strengths and areas for development. It can be used as a tool for training sessions for the squad to work on specific area. To target training to improve specific areas of performance.

What kind of data do you need to collect?

1 response

Its up to you. A sport that could be quite useful would be something like T-ball where we can measure:

Batting:
Points scored
Hit direction
Hit distance
Number of swings
Base hits
Line drives

Fielding:
Errors
Techniques used

How would you collect that data for the program?

1 response

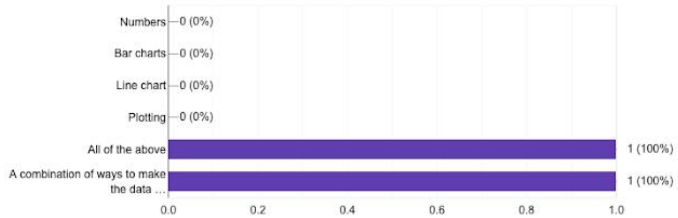
By imputing it onto a tablet or App?

What device should this program be used?

1 response

How should the data be presented?

1 response

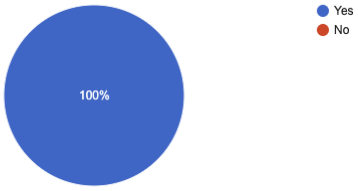


Would you like an overall rating system based on their statistics?

1 response

Would you like an overall rating system based on their statistics?

1 response



What else do you want the program to do with the data?

1 response

Keep statistics for players
Give dispersion patterns for shots
Give strengths and weakness

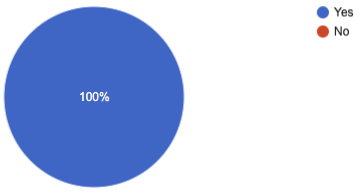
How would you want the front page and other pages to look like? (Please describe the color and the layout or send a rough sketch through gmail)

1 response

Don't mind. Something sleek and professional looking

Can you wait for one year for the app to be made?

1 response



Appendix C

Interviewer: Uhm, so my first question is do you think the app has achieved the first success criteria?

Interviewee: Okay…the first success criteria…yeah, I think it is achieved in this case. There were no errors with the validation.

Interviewer: Are there any particular feedback you would like to give for this section?

Interviewee: I think everything is fine as they are…

Interviewer: Ok-

Interviewee: Maybe for the password when signing up you could add another validation where if the password is too simple, the app would not accept. So for example, when the student input "12345678" as their password you could have the app saying "The password is too simple" and that a "Uppercase letter is needed". Something like that…

Interviewer: Yes, I'll make changes to that

Interviewee: If that makes sense?

Interviewer: Yep… uhm… so do you think the second criteria is achieved?

Interviewee: Yep, all good

Interviewer: Uhm… do you have any feedback you want to give?

Interviewee: Nope, nothing for this one

Interviewer: Ok… so now for the third success criteria do you think the third success criteria is achieved?

Interviewee: Yep, absolutely. Though maybe you should have a "forget your password" option kind of thing in there? Just in case if the students or me forget their password it would be useful to have it there so that we could sign back into our accounts again

Interviewer: Oh yeah, right ok. Got it.

Interviewee: Yeah?

Interviewer: Yep. Ok for the fourth success criteria… do you think it is achieved?

Interviewee: Yeah, maybe for the classroom ID, you could shorten it down a bit just because I think that it's a bit too lengthy and can be time consuming for the kids to just join the classroom

Interviewer: Ah ok… I'll make sure to fix that…onto the fifth success criteria… after testing out the necessary functions, do you think the fifth success criteria is achieved?

Interviewee: Yes, absolutely… I'm amazed with how well the inputs are layed out and just the interface in general. It allows just allows me to manipulate data easily. Just in general, I think it's very easy to navigate around.

Interviewer: Ok, so no improvements needed for this section of the app right?

Interviewee: Yep

Interviewer: Alright so… do you think the sixth success criteria is achieved?

Interviewee: I'd say…not fully achieved. I liked the fact that you have considered the sport's rules within the validation but one thing that may be missing from that is if suppose that you have 0 base ran then you are probably caught out or have 3 strikes and time in seconds is supposed to be nothing. So I think that once you include this then this success criteria would be achieved.

Interviewer: Ah I see, I'll try to add that validation in.

Interviewee: Yep

Interviewer: Ok, so moving onto the next success criteria… is success criteria seventh achieved?

Interviewee: Absolutely, I can see the different graphs perfectly here and its filter function works fine as well. Maybe, it could be better if the app could then have a forecasting function for the graphs? It might help me to see whether the kids may improve or not so I can see if there's a problem or not and support them. Also, you should add some information on what the different colored dots tell so that I can understand what I am looking at because currently, you would need to explain that to me and I might forget.

Interviewer: Yep ok, I'll try my best to implement those features. Uhm.. so for the eighth success criteria… do you think it is achieved?

Interviewee: Yeah, I can see the feedback I just entered within the student's account so it's achieved.

Interviewer: Ok… so finally, for the ninth success critieria, do you think it is achieved?

Interviewee: Yep, I see all the data I inputted on here, the classroom's ID and so on… yep, the ninth success criteria is definitely achieved

Interviewer: Alright, this concludes the interview.

Appendix D

main.dart

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/authenticate/register.dart';
import 'package:cstballprogram/screens/authenticate/sign_in.dart';
import 'package:cstballprogram/Redirect.dart';
import 'package:cstballprogram/services/auth.dart';
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:provider/provider.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StreamProvider<cUser?>.value(
      initialData: null,
      value: AuthService().loginstatus, // stream you want to listen to
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        home: Wrapper(),
        routes: {
          '/sign_in': (context) => SignIn(),
          '/register': (context) => Register(),
        },
      ),
    );
  }
}
```

authenticate.dart

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

bool teacher = false;

class Authenticate extends StatefulWidget {
  @override
  _AuthenticateState createState() => _AuthenticateState();
}

class _AuthenticateState extends State<Authenticate> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[900],
      body: Center(
        child: Column(
          children: [
            SizedBox(height: 10.0),
            SafeArea(
              child: Container(
                child: Text(
                  'Swing',
                  style: GoogleFonts.lato(
                    textStyle: TextStyle(
                      color: Colors.amber,
                      fontSize: 50,
                      letterSpacing: 2,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              ),
            ),
          ),
          SizedBox(height: 200.0),
          Container(
```

```
    height: 50,
    width: 300,
    child: TextButton(
      style: ButtonStyle(
        backgroundColor:
            MaterialStateProperty.all<Color>(Colors.amber),
        shape: MaterialStateProperty.all<RoundedRectangleBorder>(
            RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(18.0),
        )),
      ),
      onPressed: () {
        setState(() {
          Navigator.pushNamed(context, '/sign_in');
        });
      },
      child: Text(
        "Sign In",
        style: GoogleFonts.lato(
            textStyle: TextStyle(
          color: Colors.grey[800],
          fontSize: 15.0,
          fontWeight: FontWeight.bold,
        )),
      ),
    ),
  ),
  SizedBox(height: 20.0),
  Container(
    height: 50,
    width: 300,
    child: TextButton(
      style: ButtonStyle(
        shape: MaterialStateProperty.all<RoundedRectangleBorder>(
            RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(18.0),
                side: BorderSide(color: Colors.amber))),
```

```
        ),
        onPressed: () {
          setState(() {
            teacher = false;
            Navigator.pushNamed(context, '/register');
          });
        },
        child: Text(
          "Register as Student",
          style: GoogleFonts.lato(
              textStyle: TextStyle(
            color: Colors.amber,
            fontSize: 15.0,
            fontWeight: FontWeight.bold,
          )),
        ),
      ),
    ),
    SizedBox(height: 20.0),
    Container(
      height: 50,
      width: 300,
      child: TextButton(
        style: ButtonStyle(
          shape: MaterialStateProperty.all<RoundedRectangleBorder>(
              RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(18.0),
                  side: BorderSide(color: Colors.amber))),
        ),
        onPressed: () {
          setState(() {
            teacher = true;
            Navigator.pushNamed(context, '/register');
          });
        },
        child: Text(
          "Register as Teacher",
```

```
                style: GoogleFonts.lato(
                  textStyle: TextStyle(
                color: Colors.amber,
                fontSize: 15.0,
                fontWeight: FontWeight.bold,
                )),
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

register.dart

```
import 'package:cstballprogram/screens/authenticate/authenticate.dart';
import 'package:cstballprogram/services/auth.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class Register extends StatefulWidget {
  @override
  _RegisterState createState() => _RegisterState();
}

class _RegisterState extends State<Register> {
  bool loading = false;
  String error = '';
  final AuthService _cAuth = AuthService();
  final _formkey = GlobalKey<FormState>();
  @override
  Widget build(BuildContext context) {
    String email = '';
    String password = '';
    String name = '';

    return loading == true
        ? Loading()
        : Scaffold(
            backgroundColor: Colors.grey[800],
            appBar: AppBar(
              backgroundColor: Colors.grey[900],
              elevation: 0.0,
              centerTitle: true,
              title: Text(
                teacher
                    ? 'Teacher Register to Swing'
                    : 'Student Register to Swing',
                style: GoogleFonts.lato(
```

```
          textStyle: TextStyle(
        fontWeight: FontWeight.bold,
        color: Colors.amber,
      )),
    ),
  ),
  body: Form(
    key: _formkey,
    child: Container(
      padding: EdgeInsets.symmetric(vertical: 20.0, horizontal: 50.0),
      child: Column(
        children: [
          SizedBox(height: 20),
          TextFormField(
            style: TextStyle(color: Colors.amber),
            decoration:
                textInputDecoration.copyWith(hintText: 'Email'),
            validator: (val) =>
                val!.isEmpty ? 'Enter an email' : null,
            onChanged: (val) {
              email = val;
            },
          ),
          SizedBox(height: 20),
          TextFormField(
            style: TextStyle(color: Colors.amber),
            decoration:
                textInputDecoration.copyWith(hintText: 'Password'),
            validator: (val) => val!.length < 8
                ? 'Enter a password 8+ characters long'
                : null,
            obscureText: true,
            onChanged: (val) {
              password = val;
            },
          ),
          SizedBox(
```

```
  height: 20,
),
TextFormField(
  style: TextStyle(color: Colors.amber),
  decoration:
      textInputDecoration.copyWith(hintText: 'Full Name'),
  validator: (val) =>
      val!.isEmpty ? 'Enter a Full Name' : null,
  onChanged: (val) {
    name = val;
  },
),
SizedBox(
  height: 20,
),
ElevatedButton(
  style: TextButton.styleFrom(
    backgroundColor: Colors.amber,
    fixedSize: Size(300, 30),
  ),
  onPressed: () async {
    //if it returns null from both validators, it will proceed as true
    if (_formkey.currentState!.validate()) {
      setState(() {
        loading = true;
      });
      dynamic result = await _cAuth
          .registerEmailAndPassword(name, email, password);
      if (result == null) {
        setState(() {
          error = 'Invalid email';
          loading = false;
        });
        //unable to catch error in email format
      } else {
        setState(() {
          Navigator.pop(context);
```

```
                    });
                  }
                }
              },
              child: Text(
                teacher ? 'Register as Teacher' : 'Register as Student',
                style: TextStyle(color: Colors.grey[900]),
              ),
            ),
            SizedBox(
              height: 12.0,
            ),
            Text(
              error,
              style: TextStyle(color: Colors.red, fontSize: 14.0),
            )
          ],
        ),
      ),
    ),
  );
 }
}
```

sign_in.dart

```dart
import 'package:cstballprogram/services/auth.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class SignIn extends StatefulWidget {
  @override
  _SignInState createState() => _SignInState();
}

class _SignInState extends State<SignIn> {
  final AuthService _cAuth = AuthService();
  final _formkey = GlobalKey<FormState>();
  String error = '';
  bool loading = false;
  @override
  Widget build(BuildContext context) {
    String email = '';
    String password = '';
    return loading == true
        ? Loading()
        : Scaffold(
            backgroundColor: Colors.grey[800],
            appBar: AppBar(
              backgroundColor: Colors.grey[900],
              elevation: 0.0,
              title: Text(
                'Sign in to Swing',
                style: GoogleFonts.lato(
                    textStyle: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Colors.amber,
                )),
              ),
            ),
```

```
      body: Container(
        padding: EdgeInsets.symmetric(vertical: 20.0, horizontal: 50.0),
        child: Form(
          key: _formkey,
          child: Column(
            children: [
              SizedBox(height: 20),
              TextFormField(
                style: TextStyle(color: Colors.amber),
                decoration:
                    textInputDecoration.copyWith(hintText: 'Email'),
                validator: (val) =>
                    val!.isEmpty ? 'Enter an email' : null,
                onChanged: (val) {
                  email = val;
                },
              ),
              SizedBox(height: 20),
              TextFormField(
                style: TextStyle(color: Colors.amber),
                decoration:
                    textInputDecoration.copyWith(hintText: 'Password'),
                validator: (val) =>
                    val!.isEmpty ? 'Enter a password' : null,
                obscureText: true,
                onChanged: (val) {
                  password = val;
                },
              ),
              SizedBox(
                height: 20,
              ),
              ElevatedButton(
                style: TextButton.styleFrom(
                  backgroundColor: Colors.amber,
                ),
                onPressed: () async {
```

```
                    if (_formkey.currentState!.validate()) {
                      setState(() {
                        loading = true;
                      });
                      dynamic result = await _cAuth.signInEmailAndPassword(
                          email, password);
                      if (result == null) {
                        setState(() {
                          error = 'Could not Sign In';
                          loading = false;
                        });
                      } else {
                        setState(() {
                          Navigator.pop(context);
                        });
                      }
                    }
                  },
                  child: Text(
                    'Sign in',
                    style: TextStyle(color: Colors.grey[900]),
                  ),
                ),
                SizedBox(
                  height: 12.0,
                ),
                Text(
                  error,
                  style: TextStyle(color: Colors.red, fontSize: 14.0),
                )
              ],
            ),
          ),
        ),
      );
  }
}
```

Redirect.dart

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/authenticate/authenticate.dart';
import 'package:cstballprogram/screens/student/home/StudentHome.dart';
import 'package:cstballprogram/screens/teacher/home/TeacherHome.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

class Wrapper extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final userloginuid = Provider.of<cUser?>(context);

    //return either home or authenticate
    if (userloginuid == null) {
      return Authenticate();
    } else {
      return StreamBuilder<UserRole?>(
          stream: DatabaseService(uid: userloginuid.uid.toString()).userrole,
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return Loading();
            }
            if (snapshot.data?.teacher == true) {
              return Home();
            }
            if (snapshot.data?.teacher == false) {
              return SHome();
            }
            return Loading();
          });
    }
  }
}
```

TeacherHome.dart (*Teacher Screen)*

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/teacher/classroom/SelectedClass.dart';
import 'package:cstballprogram/services/auth.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:provider/provider.dart';

class Home extends StatelessWidget {
  final AuthService _auth = AuthService();

  @override
  Widget build(BuildContext context) {
    final loginuseruid = Provider.of<cUser?>(context);

    return MultiProvider(
        providers: [
          StreamProvider<TeacherData?>.value(
            value: DatabaseService(uid: loginuseruid?.uid).teacherdata,
            initialData: null,
            catchError: (context, error) => null,
          ),
          StreamProvider<QuerySnapshot?>.value(
            value: DatabaseService().classroomCollectionData,
            initialData: null,
            catchError: (context, error) => null,
          ),
        ],
        child: Scaffold(
          backgroundColor: Colors.grey[400],
          appBar: AppBar(
            title: Text(
              'Swing',
```

```
        style: TextStyle(
          color: Colors.amber,
        ),
      ),
      centerTitle: false,
      backgroundColor: Colors.grey[800],
      elevation: 0.0,
      actions: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: TextButton.icon(
            style: ButtonStyle(
              backgroundColor:
                  MaterialStateProperty.all<Color>(Colors.amber),
              shape: MaterialStateProperty.all<RoundedRectangleBorder>(
                  RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(6.0),
              )),
            ),
            onPressed: () async {
              await _auth.logOut();
            },
            icon: Icon(
              Icons.person,
              color: Colors.grey[800],
            ),
            label: Text('Logout',
                style: GoogleFonts.lato(
                  textStyle:
                      TextStyle(color: Colors.grey[800], fontSize: 15.0),
                )),
          ),
        )
      ],
    ),
    body: ClassroomList(),
  ));
```

```
  }
}

class ClassroomList extends StatefulWidget {
  @override
  _ClassroomListState createState() => _ClassroomListState();
}

class _ClassroomListState extends State<ClassroomList> {
  bool loading = false;
  bool samedata = false;
  String inputname = '';
  String error = '';
  final _formkey = GlobalKey<FormState>();

  //Pop-up for adding new classroom
  Future<void> newClassroomName(
      BuildContext context,
      var classlist,
      String defaultname,
      String? uid,
      String ID,
      int numofclass,
      String teachername) async {
    inputname = defaultname;
    return await showDialog(
        context: context,
        builder: (context) {
          return StatefulBuilder(builder: (context, setState) {
            return Form(
              key: _formkey,
              child: AlertDialog(
                title: Text("Enter Class Name Below"),
                content: Column(
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    TextFormField(
```

```
          initialValue: defaultname,
          style: TextStyle(color: Colors.amber),
          validator: (val) =>
              val!.isEmpty ? 'Please enter a name' : null,
          decoration:
              textInputDecoration.copyWith(hintText: 'Class Name'),
          onChanged: (val) {
            inputname = val;
          },
        ),
        SizedBox(
          height: 20,
        ),
        Text(
          error,
          style: TextStyle(color: Colors.red, fontSize: 14.0),
        )
      ],
    ),
    actions: [
      TextButton(
          onPressed: () async {
            if (_formkey.currentState!.validate()) {
              setState(() {
                samedata = false;
              });
              for (var x in classlist) {
                if (inputname == x[0]) {
                  setState(() {
                    error = 'Classroom Name Already Exist!';
                    samedata = true;
                  });
                }
              }
              if (samedata == false) {
                await DatabaseService(uid: uid)
                    .addTNewClass(ID, numofclass);
```

```
                    await DatabaseService().createClassroomData(
                        ID, [], inputname, teachername);
                    Navigator.pop(context);
                  }
                }
              },
              child: Text("Create New Classroom"))
          ],
        ),
      );
    });
  });
}

@override
Widget build(BuildContext context) {
  final loginuseruid = Provider.of<cUser?>(context);
  final teacher = Provider.of<TeacherData?>(context);
  final classroomdataquery = Provider.of<QuerySnapshot?>(context);

  final classroomdatadoc = classroomdataquery?.docs;

  var classidlist = teacher?.classid == null ? [] : teacher?.classid;
  var student = List.generate(
      classidlist.length, (i) => List.filled(0, [], growable: true),
      growable: true); // "student" is a dynamic multi-dimensional arrays
  var class2dlist = [];
  int index = 0;

  if (classroomdatadoc != null) {
    for (var x in classroomdatadoc) {
      for (var y = 0; y < classidlist.length; y++) {
        if (x.get('id') == classidlist[y]) {
          /* if teacher's classroom id ("classidlist[y]") matches with
          a classroom id stored in the database ("x.get('id')") then add to array "class2dlist"*/
          class2dlist.add([
            x.get('classname'),
```

```
            classidlist[y]
          ]); //class2dlist: [[class name, class id], [..., ...]]
          var studentidlist = x.get(
              'studentid'); // "studentidlist" contains the id of students that are within the classroom
          for (var z in studentidlist) {
            //converting from an array of maps to an array of arrays for ease of access
            student[index].add([
              z.keys.toString().substring(1, z.keys.toString().length - 1),
              z.values.toString().substring(1, z.values.toString().length - 1)
            ]);
          }
          index++; //to move onto the next "slot" everytime student data is added to dynamic multi-dimensional array
"student",
        }
      }
    }
  }

  String? teachername = teacher?.name;
  int? numofclass = teacher?.numofclass;
  final String ID = new DateTime.now().millisecondsSinceEpoch.toString();

  if (teacher == null || classroomdataquery == null || loading == true) {
    return Loading();
  } else {
    return Stack(
      children: [
        ListView.builder(
            itemCount: class2dlist.length,
            itemBuilder: (BuildContext context, int index) {
              return Padding(
                padding: EdgeInsets.only(top: 8.0),
                child: GestureDetector(
                  child: Card(
                    margin: EdgeInsets.fromLTRB(20.0, 6.0, 20.0, 0.0),
                    child: ListTile(
                      leading: CircleAvatar(
```

```
          radius: 25.0,
          child: Icon(Icons.sports_baseball),
        ),
        trailing: Container(
          height: 40,
          width: 40,
          child: FloatingActionButton(
            heroTag: "deletebtn$index",
            elevation: 0,
            foregroundColor: Colors.red,
            backgroundColor: Colors.white,
            child: Icon(Icons.delete),
            onPressed: () {
              showDialog(
                  context: context,
                  builder: (context) => AlertDialog(
                        title: Text("Delete?"),
                        content: Text(
                            "Classroom will be permanently deleted."),
                        actions: [
                          TextButton(
                              onPressed: () async {
                                setState(() {
                                  loading = true;
                                });
                                await DatabaseService(
                                        uid: loginuseruid!.uid)
                                    .deleteCIDfromUserCList(
                                        class2dlist[index][1]
                                            .toString());
                                if (student[index].length !=
                                    0) {
                                  for (var studentid
                                      in student[index]) {
                                    await DatabaseService(
                                            uid: studentid[0]
                                                .toString())
```

```
                                .deleteCIDfromUserCList(
                                    class2dlist[index]
                                        [1]
                                    .toString());
                              }
                            }

                            await DatabaseService()
                                .deleteClassroomDocument(
                                    class2dlist[index][1]
                                        .toString());
                            setState(() {
                              loading = false;
                            });
                            Navigator.pop(context);
                          },
                          child: Text("Yes")),
                      TextButton(
                          onPressed: () {
                            Navigator.pop(context);
                          },
                          child: Text("No"))
                    ],
                  ));
        },
      ),
    ),
    title: Text('${class2dlist[index][0]}'),
    subtitle: Text("ID: ${class2dlist[index][1]}"),
  ),
),
onTap: () async {
  Navigator.of(context).push(MaterialPageRoute(
      builder: (context) => TeacherClassroom(
          name: class2dlist[index][0],
          id: class2dlist[index][1],
          student: student[
```

```
                        index], //Student's info in the classroom
                    )));
          },
        ),
      );
    }),
  Positioned(
    bottom: 30,
    right: 30,
    child: Container(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Container(
            height: 70,
            width: 70,
            child: FloatingActionButton(
              foregroundColor: Colors.grey[800],
              backgroundColor: Colors.amber,
              onPressed: () async {
                await newClassroomName(
                    context,
                    class2dlist,
                    "New Classroom $numofclass",
                    loginuseruid!.uid,
                    ID,
                    numofclass!,
                    teachername!);
              },
              child: Icon(
                Icons.add,
              ),
            ),
          ),
        ],
      ),
```

```
          ),
        ),
      ],
    );
  }
 }
}
```

SelectedClass.dart (*Teacher Screen*)

```dart
import 'package:cstballprogram/models/classroom.dart';
import 'package:cstballprogram/screens/teacher/classroom/ViewStudentStat.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';

class TeacherClassroom extends StatefulWidget {
  final String? name;
  final String? id;
  var student;
  TeacherClassroom({this.name, this.id, this.student});

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

class _TeacherClassroomState extends State<TeacherClassroom> {
  bool loading = false;
  @override
  Widget build(BuildContext context) {
    if (widget.name == null ||
        widget.id == null ||
        widget.student == null ||
        loading == true) {
      return Loading();
    }
    return StreamBuilder<ClassDocData>(
        stream: DatabaseService(ID: widget.id).classroomDocumentData,
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            var studentList = snapshot.data!.studentid;
            String? classname = snapshot.data!.classname;
            return Scaffold(
              backgroundColor: Colors.grey[400],
              appBar: AppBar(
                title: Text('${classname}'),
```

```
  backgroundColor: Colors.grey[800],
),
body: widget.student.length != 0
    ? ListView.builder(
        itemCount: studentList.length,
        itemBuilder: (context, index) {
          return Padding(
            padding: EdgeInsets.only(top: 8.0),
            child: GestureDetector(
              child: Card(
                margin: EdgeInsets.fromLTRB(20.0, 6.0, 20.0, 0.0),
                child: ListTile(
                  trailing: Container(
                    height: 40,
                    width: 40,
                    child: FloatingActionButton(
                      heroTag: "deletebtn$index",
                      elevation: 0,
                      foregroundColor: Colors.red,
                      backgroundColor: Colors.white,
                      child: Icon(Icons.delete),
                      onPressed: () {
                        showDialog(
                            context: context,
                            builder: (context) => AlertDialog(
                                title: Text("Delete?"),
                                content: Text(
                                    "Student will be permanently removed."),
                                actions: [
                                  TextButton(
                                      onPressed: () async {
                                        String studentid = studentList[
                                                index]
                                            .keys
                                            .toString()
                                            .substring(
                                                1,
```

```
        studentList[index]
                .keys
                .toString()
                .length -
            1);
String studentname = studentList[
        index]
    .values
    .toString()
    .substring(
        1,
        studentList[index]
                .values
                .toString()
                .length -
            1);
setState(() {
  loading = true;
});
if (widget.student
        .length !=
    0) {
  await DatabaseService(
          uid:
              studentid)
      .deleteCIDfromUserCList(
          widget.id
              .toString());
  await DatabaseService(
          ID: widget.id)
      .deleteStudentFromClassroomList(
          studentid,
          studentname);
}
setState(() {
  loading = false;
});
```

```
                                    Navigator.pop(context);
                                  },
                                  child: Text("Yes")),
                              TextButton(
                                  onPressed: () {
                                    Navigator.pop(context);
                                  },
                                  child: Text("No"))
                            ],
                          ));
                    },
                  ),
                ),
                leading: CircleAvatar(
                  radius: 25.0,
                  child: Icon(Icons.sports_baseball),
                ),
                title: Text(
                  '${studentList[index].values.toString().substring(1, studentList[index].values.toString().length -
1)}'),
              ),
            ),
            onTap: () {
              Navigator.of(context).push(MaterialPageRoute(
                  builder: (context) => TeacherStudentStat(
                      studentname:
                          '${widget.student[index][1]}',
                      studentid:
                          '${widget.student[index][0]}')));
            },
          ),
        );
      })
    : null,
  );
} else {
  return Loading();
```

```
        }
      });
  }
}
```

ViewStudentStats.dart (*Teacher Screen*)

```
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/teacher/edit/EditFeedback.dart';
import 'package:cstballprogram/screens/teacher/edit/EditInputData.dart';
import 'package:cstballprogram/screens/teacher/input/InputFeedback.dart';
import 'package:cstballprogram/screens/teacher/input/InputData.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:cstballprogram/shared/stats.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

class TeacherStudentStat extends StatefulWidget {
  String? studentname;
  String? studentid;
  TeacherStudentStat({this.studentname, this.studentid});
  @override
  _TeacherStudentStatState createState() => _TeacherStudentStatState();
}

class _TeacherStudentStatState extends State<TeacherStudentStat> {
  String? page = '';

  @override
  Widget build(BuildContext context) {
    if (widget.studentname == null || widget.studentid == null) {
      return Loading();
    }
    return StreamProvider<StudentData?>.value(
      value: DatabaseService(uid: widget.studentid).studentdata,
      catchError: (context, error) => null,
      initialData: null,
      child: Scaffold(
        backgroundColor: Colors.grey[400],
        appBar: AppBar(
          title: Text('${widget.studentname}'),
          centerTitle: true,
```

```
backgroundColor: Colors.grey[800],
actions: [
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: DropdownButton(
      dropdownColor: Colors.grey[800],
      underline: Container(
        color: Colors.transparent,
      ),
      onChanged: (String? pageinput) {
        if (pageinput == 'Input Data') {
          Navigator.push(
              context,
              MaterialPageRoute(
                  builder: (context) =>
                      StreamProvider<StudentData?>.value(
                          value:
                              DatabaseService(uid: widget.studentid)
                                  .studentdata,
                          initialData: null,
                          catchError: (context, error) => null,
                          child: TeacherInputStudentData(
                            studentid: widget.studentid,
                          ))));
        }
        if (pageinput == 'Input Feedback') {
          Navigator.push(
              context,
              MaterialPageRoute(
                  builder: (context) =>
                      StreamProvider<StudentData?>.value(
                          value:
                              DatabaseService(uid: widget.studentid)
                                  .studentdata,
                          initialData: null,
                          catchError: (context, error) => null,
                          child: TeacherFeedback(
```

```
                    studentid: widget.studentid,
                  )));
    }
    if (pageinput == 'Edit Input Data') {
      Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => EditInputData(
                  studentname: widget.studentname,
                  studentid: widget.studentid,
                )));
    }
    if (pageinput == 'Edit Feedback') {
      Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => EditFeedback(
                  studentid: widget.studentid,
                  studentname: widget.studentname,
                )));
    }
  },
  items: [
    'Input Data',
    'Input Feedback',
    'Edit Input Data',
    "Edit Feedback"
  ].map((String value) {
    return DropdownMenuItem(
      value: value,
      child: Text(value),
    );
  }).toList(),
  icon: Icon(
    Icons.menu_rounded,
    color: Colors.amber,
    size: 35.0,
```

```
              ),
              style: TextStyle(color: Colors.amber),
            ),
          ),
        ],
      ),
      body: Stats(),
    ),
  );
 }
}
```

InputData.dart (*Teacher Screen*)

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'dart:math';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';

// ignore: must_be_immutable
class TeacherInputStudentData extends StatefulWidget {
  String? studentid;
  var dataList;
  var dateOfInputList;
  var dataInputList;
  var dataInputSelect;
  var index;
  TeacherInputStudentData(
      {this.studentid,
      this.dataList,
      this.dateOfInputList,
      this.dataInputList,
      this.dataInputSelect,
      this.index});
  @override
  _TeacherInputStudentDataState createState() =>
      _TeacherInputStudentDataState();
}

class _TeacherInputStudentDataState extends State<TeacherInputStudentData> {
  bool loading = false;
  final _formkey = GlobalKey<FormState>();
  int noofstrike = 0;
  double seconds = 0;
  int noofbaseran = 0;
  double distance = 0;
```

```
String caughtout = '';
double posx = 100.0;
double posy = 100.0;
bool changecolor = false;
String error = '';
double newaccuracy = 0.0;
double newspeed = 0.0;
double neweffect = 0.0;
double newdistance = 0.0;
double avgaccuracy = 0.0;
double? avgspeed = 0.0;
double avgdistance = 0.0;
double avgeffect = 0.0;
String? dateinput;
int count = 0;

void onTapDown(BuildContext context, TapDownDetails details) {
  setState(() {
    posx = details.localPosition.dx;
    posy = details.localPosition.dy;
    distance = double.parse(sqrt(
            pow(((379 - posx) * (1 / 4)), 2) + pow((512 - posy) * (1 / 4), 2))
        .toStringAsFixed(1));
    changecolor = true;
  });
}

void onDoubleTap() {
  setState(() {
    posx = 100.0;
    posy = 100.0;
    distance = 0.0;
    changecolor = false;
  });
}

@override
```

```
 Widget build(BuildContext context) {
   final studentdata = Provider.of<StudentData?>(context);

   //This is to allow the client to change data as it prevents the variables from returning to the original data when
changing
   if (count < 1) {
     if (widget.dataList != null) {
       noofstrike = int.parse((widget.dataInputSelect[0]));
       seconds = double.parse(widget.dataInputSelect[1]);
       noofbaseran = int.parse(widget.dataInputSelect[2]);
       caughtout = widget.dataInputSelect[3].trim();
       posx = double.parse(widget.dataInputSelect[4]);
       posy = double.parse(widget.dataInputSelect[5]);
       distance = double.parse(sqrt(pow(((379 - posx) * (1 / 4)), 2) +
             pow((512 - posy) * (1 / 4), 2))
           .toStringAsFixed(1));
       changecolor = true;
       count++;
     }
   }

   final mapacc = studentdata?.accList;
   final mapspeed = studentdata?.speedList;
   final mapdistance = studentdata?.distanceList;
   final mapeffect = studentdata?.effectList;
   final mapdatainput = studentdata?.datainput;

   double acctotal = 0.0;
   double speedtotal = 0.0;
   double distancetotal = 0.0;
   double effecttotal = 0.0;
   int noofspeed = 0;

   //Totalling for averages
   if (mapacc != null &&
       mapspeed != null &&
       mapdistance != null &&
```

```
      mapeffect != null) {
    for (var z = 0; z < mapacc.length; z++) {
      //totalling for accuracy
      acctotal += double.parse(mapacc[z]
          .values
          .toString()
          .substring(1, mapacc[z].values.toString().length - 1));
      //totalling for distance
      distancetotal += double.parse(mapdistance[z]
          .values
          .toString()
          .substring(1, mapdistance[z].values.toString().length - 1));
      //totalling for effectiveness
      effecttotal += double.parse(mapeffect[z]
          .values
          .toString()
          .substring(1, mapeffect[z].values.toString().length - 1));
      //If they were eliminated before they ran, speed = -314 and therefore it doesn't count in the totalling and
averages
      if (double.parse(mapspeed[z]
              .values
              .toString()
              .substring(1, mapspeed[z].values.toString().length - 1)) !=
          -314) {
        speedtotal += double.parse(mapspeed[z]
            .values
            .toString()
            .substring(1, mapspeed[z].values.toString().length - 1));
        noofspeed++;
      }
    }
  }

  return loading == true
      ? Loading()
      : Scaffold(
          backgroundColor: Colors.grey[400],
```

```
      appBar: AppBar(
        title: widget.dataList != null
            ? Text('Edit data')
            : Text('Input data'),
        backgroundColor: Colors.grey[800],
        elevation: 0.0,
        actions: [
          //The "Add/Change" Button
          ElevatedButton(
            child: widget.dataList != null
                ? Text('Change',
                    style: TextStyle(color: Colors.grey[900]))
                : Text(
                    'Add',
                    style: TextStyle(color: Colors.grey[900]),
                  ),
            style: TextButton.styleFrom(
                backgroundColor: Colors.amber,
                padding: EdgeInsets.all(8.0)),
            onPressed: () async {
              if (_formkey.currentState!.validate()) {
                //Checks if there is input position of ball
                if (changecolor == false) {
                  setState(() {
                    loading = false;
                    error = "Please enter the position of the ball below";
                  });
                } else {
                  //Checks if the ball is land in the correct spots for a hit to count
                  if (noofstrike < 3 &&
                      (distance < 14.8 ||
                          512 - posy < 0 ||
                          379 - posx < 0)) {
                    setState(() {
                      loading = false;
                      error =
                          "A proper hit cannot land there at $noofstrike ${noofstrike == 2 ? 'strikes' : 'strike'}!";
```

```
    });
  } else {
    /*
    Calculation from input section
    */

    //calculates accuracy
    newaccuracy = double.parse(
        (1 - noofstrike / 3).toStringAsFixed(2));
    //If seconds = 0 then can't calculate speed and assign it to -314
    newspeed = seconds == 0
        ? -314
        : double.parse(((noofbaseran * 60) / seconds)
            .toStringAsFixed(1));
    //Assigns effectiveness based on the no. of base ran
    if (noofbaseran != 4 && noofbaseran != 0) {
      if (caughtout == 'Y') {
        if (noofbaseran == 1) {
          neweffect = 0.15;
        }
        if (noofbaseran == 2) {
          neweffect = 0.45;
        }
        if (noofbaseran == 3) {
          neweffect = 0.75;
        }
      } else {
        if (noofbaseran == 1) {
          neweffect = 0.30;
        }
        if (noofbaseran == 2) {
          neweffect = 0.60;
        }
        if (noofbaseran == 3) {
          neweffect = 0.90;
        }
      }
```

```
} else {
  if (noofbaseran == 4) {
    neweffect = 1;
  } else {
    neweffect = 0;
  }
}
//Assigns 'distance' to 'newdistance'
newdistance = distance;

/*
Average Calculation Section
*/

//Calculates new average accuracy
avgaccuracy = double.parse(((acctotal + newaccuracy) /
        (mapacc.length +
            1 -
            count)) //Since if there is no change, n data would not increase by one
    .toStringAsFixed(2));
//Calculates new average speed only if > 0
if (newspeed != -314) {
  avgspeed = double.parse(((speedtotal + newspeed) /
          (noofspeed + 1 - count))
      .toStringAsFixed(1));
} else {
  avgspeed = studentdata?.speed;
}
//Calculates average distance
avgdistance = double.parse(
    ((distancetotal + newdistance) /
            (mapdistance.length + 1 - count))
        .toStringAsFixed(1));
//Calculates average effect
avgeffect = double.parse(((effecttotal + neweffect) /
        (mapeffect.length + 1 - count))
    .toStringAsFixed(2));
```

```
/*
Date, Storing Data Input, and Finalizing data section
*/

//Stores date input
dateinput = widget.dataList != null
    ? widget.dateOfInputList[widget.index]
    : DateFormat('yyyyMMdd').format(DateTime.now());

//Adds new data inputted or edit the data
if (widget.dataList == null) {
  mapacc.add({dateinput: newaccuracy});
  mapspeed.add({dateinput: newspeed});
  mapdistance.add({dateinput: newdistance});
  mapeffect.add({dateinput: neweffect});
  mapdatainput.add({
    dateinput: [
      noofstrike,
      seconds,
      noofbaseran,
      caughtout,
      double.parse((posx).toStringAsFixed(2)),
      double.parse((posy).toStringAsFixed(2))
    ]
  });
} else {
  mapacc[widget.index] = ({dateinput: newaccuracy});
  mapspeed[widget.index] = ({dateinput: newspeed});
  mapdistance[widget.index] =
      ({dateinput: newdistance});
  mapeffect[widget.index] = ({dateinput: neweffect});
  mapdatainput[widget.index] = ({
    dateinput: [
      noofstrike,
      seconds,
      noofbaseran,
```

```
                caughtout,
                double.parse((posx).toStringAsFixed(2)),
                double.parse((posy).toStringAsFixed(2))
              ]
            });
          }
          //Updates new data inputted to the database
          await DatabaseService(uid: widget.studentid)
              .updateStudentStatsFromInput(
                mapacc,
                mapspeed,
                mapeffect,
                mapdistance,
                avgaccuracy,
                avgspeed!,
                avgeffect,
                avgdistance,
                mapdatainput);
          Navigator.pop(context);
        }
      }
    }
  },
 ),
],
),
body: Column(
 children: [
  Form(
   key: _formkey,
   child: Row(
    children: [
     Expanded(
      child: Container(
       padding: EdgeInsets.symmetric(
          vertical: 20.0, horizontal: 20.0),
       color: Colors.grey[800],
```

```
            child: Column(
              children: [
                ConstrainedBox(
                  constraints: BoxConstraints(minHeight: 78.0),
                  child: TextFormField(
                    initialValue: widget.dataList != null
                        ? widget.dataInputSelect[0]
                        : null,
                    style: TextStyle(color: Colors.amber),
                    decoration: textInputDecoration.copyWith(
                        hintText: 'No.of Strikes'),
                    onChanged: (val) {
                      try {
                        int.parse(val);
                        setState(() {
                          noofstrike = int.parse(val);
                          if (noofstrike == 3) {
                            seconds = 0.0;
                            noofbaseran = 0;
                            caughtout = 'N';
                          }
                        });
                      } catch (e) {
                        setState(() {
                          noofstrike = 0;
                        });
                      }
                    },
                    validator: (val) {
                      if (val!.isNotEmpty) {
                        try {
                          noofstrike = int.parse(val);
                          if (noofstrike < 0 ||
                              noofstrike > 3) {
                            return 'Please Enter a No. from 0-3';
                          }
                        } catch (e) {
```

```
            return 'Please Input an Integer!';
          }
        } else {
          return "Please Enter a Value!";
        }
      }),
),
SizedBox(height: 5.0),
//Text Form Field for Running Time
ConstrainedBox(
  constraints: BoxConstraints(minHeight: 78.0),
  child: TextFormField(
    initialValue: widget.dataList != null
        ? widget.dataInputSelect[1]
        : null,
    readOnly: noofstrike == 3 ? true : false,
    style: TextStyle(color: Colors.amber),
    decoration: textInputDecoration.copyWith(
      hintText: noofstrike == 3
          ? 'N/A'
          : 'Time in seconds',
      hintStyle: noofstrike == 3
          ? TextStyle(color: Colors.amber)
          : TextStyle(
              color: Color(0xFFBDBDBD))),
    onChanged: (val) {
      try {
        double.parse(val);
        seconds = double.parse(val);
      } catch (e) {}
    },
    validator: (val) {
      if (noofstrike != 3) {
        if (val!.isNotEmpty) {
          try {
            double.parse(val);
            if (double.parse(val) <= 0) {
```

```
                        return 'Input a time greater than 0';
                      }
                    } catch (e) {
                      return 'Please Input an Number!';
                    }
                  } else {
                    return "Please Enter a Value!";
                  }
                }
              }),
          ),
        ],
      ),
    ),
  ),
),
Expanded(
  child: Container(
    padding: EdgeInsets.symmetric(
        vertical: 20.0, horizontal: 20.0),
    color: Colors.grey[800],
    child: Column(
      children: [
        ConstrainedBox(
          constraints: BoxConstraints(minHeight: 78.0),
          child: TextFormField(
              initialValue: widget.dataList != null
                  ? widget.dataInputSelect[2]
                  : null,
              readOnly: noofstrike == 3 ? true : false,
              style: TextStyle(color: Colors.amber),
              decoration: textInputDecoration.copyWith(
                  hintText: noofstrike == 3
                      ? '0'
                      : 'No.of Base Ran',
                  hintStyle: noofstrike == 3
                      ? TextStyle(color: Colors.amber)
                      : TextStyle(
```

```
                color: Color(0xFFBDBDBD))),
      onChanged: (val) {
        try {
          int.parse(val);
          noofbaseran = int.parse(val);
        } catch (e) {}
      },
      validator: (val) {
        if (noofstrike != 3) {
          if (val!.isNotEmpty) {
            try {
              noofbaseran = int.parse(val);
              if (noofbaseran < 0 ||
                  noofbaseran > 4) {
                return 'Please Enter a No. from 0-4';
              }
            } catch (e) {
              return 'Please Input an Integer!';
            }
          } else {
            return "Please Enter a Value!";
          }
        }
      }),
),
SizedBox(
  height: 5.0,
),
ConstrainedBox(
  constraints: BoxConstraints(minHeight: 78.0),
  child: TextFormField(
      initialValue: widget.dataList != null
          ? widget.dataInputSelect[3]
          : null,
      readOnly: noofstrike == 3 ? true : false,
      style: TextStyle(color: Colors.amber),
      decoration: textInputDecoration.copyWith(
```

```
                        hintText: noofstrike == 3
                            ? 'N'
                            : 'Caught Out?',
                        hintStyle: noofstrike == 3
                            ? TextStyle(color: Colors.amber)
                            : TextStyle(
                                color: Color(0xFFBDBDBD))),
                    onChanged: (val) {
                      caughtout = val.trim();
                    },
                    validator: (val) {
                      if (noofstrike != 3) {
                        if (val!.isNotEmpty) {
                          if (val.trim() != 'Y' &&
                              val.trim() != 'N') {
                            return "Please enter either (Y/N)";
                          }
                        } else {
                          return "Please enter either (Y/N)";
                        }
                      }
                    }),
              ),
            ],
          ),
        ),
      ),
    ],
  ),
),
Expanded(
    flex: 1,
    child: Container(
      width: double.infinity,
      color: Colors.grey[800],
      child: Column(
        children: [
```

```
            Text(
              error,
              style: TextStyle(color: Colors.red, fontSize: 15.0),
            ),
            SizedBox(
              height: 5.0,
            ),
            Divider(
              color: Colors.grey[400],
              indent: 8.0,
              endIndent: 8.0,
              height: 8.0,
              thickness: 3.0,
            ),
            SizedBox(height: 5.0),
            Center(
              child: Text(
                "Input Position of Ball Below by Tapping",
                style: TextStyle(
                    color: Colors.amber, fontSize: 15.0),
              ),
            ),
          ],
        ),
      )),
    Expanded(
      flex: 10,
      child: GestureDetector(
        onTapDown: (TapDownDetails details) =>
            onTapDown(context, details),
        onDoubleTap: () => onDoubleTap(),
        child: Stack(
          children: [
            Container(
              color: Colors.green,
            ),
            Padding(
```

```
  padding: EdgeInsets.only(right: 45, bottom: 50),
  child: Align(
    alignment: Alignment.bottomRight,
    child: Container(
      height: 475,
      width: 5,
      color: Colors.white,
    ),
  ),
),
Padding(
  padding: EdgeInsets.only(bottom: 50, right: 40),
  child: Align(
    alignment: Alignment.bottomCenter,
    child: Container(
      height: 5,
      width: 375,
      color: Colors.white,
    ),
  ),
),
Padding(
  padding: const EdgeInsets.only(right: 45, bottom: 50),
  child: Align(
    alignment: Alignment.bottomRight,
    child: Container(
      height: 240,
      width: 240,
      decoration: BoxDecoration(
          border: Border.all(
              color: Colors.white, width: 5)),
      child: Padding(
        padding: const EdgeInsets.all(102.0),
        child: ClipOval(
          child: Container(
            color: Colors.brown[600],
          ),
```

```
          ),
        ),
      ),
    ),
  ),
  Padding(
    padding:
        const EdgeInsets.only(bottom: 293, right: 290),
    child: Align(
      alignment: Alignment.bottomRight,
      child: Container(
        width: 100,
        height: 100,
        child: CustomPaint(
          painter: OpenPainter(),
        ),
      ),
    ),
  ),
  Padding(
    padding:
        const EdgeInsets.only(right: 45, bottom: 275),
    child: Align(
        alignment: Alignment.bottomRight, child: bases),
  ),
  Padding(
    padding:
        const EdgeInsets.only(right: 270, bottom: 275),
    child: Align(
        alignment: Alignment.bottomRight, child: bases),
  ),
  Padding(
    padding:
        const EdgeInsets.only(right: 270, bottom: 50),
    child: Align(
        alignment: Alignment.bottomRight, child: bases),
  ),
```

```
            Positioned(
              child: Container(
                height: 50,
                width: 100,
                child: Column(
                  children: [
                    Text(
                      "$distance ft",
                      style: TextStyle(
                          color: changecolor == false
                              ? Colors.green
                              : Colors.black,
                          fontSize: 13.0),
                    ),
                    SizedBox(height: 5.0),
                    ClipOval(
                      child: Container(
                        height: 20,
                        width: 20,
                        color: changecolor == false
                            ? Colors.green
                            : Colors.red,
                      ),
                    ),
                  ],
                ),
              ),
              left: posx - 50,
              top: posy - 28,
            ),
          ],
        ),
      )
    ],
  ),
);
```

```
  }
}

class OpenPainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    var paint1 = Paint()
      ..color = Color(0xFFFFFFFF)
      ..style = PaintingStyle.stroke
      ..strokeWidth = 5;
    //draw arc
    canvas.drawArc(
        Offset(0, 0) & Size(400, 400),
        (3 * pi) / 4, //radians
        pi, //radians
        false,
        paint1);
    canvas.drawArc(
        Offset(280.8, 280.8) & Size(118.4, 118.4),
        pi, //radians
        pi / 2, //radians
        false,
        paint1);
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) => true;
}
```

InputFeedback.dart (*Teacher Screen*)

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';

class TeacherFeedback extends StatefulWidget {
  String? studentid = '';
  var feedbackMap;
  var feedbackList;
  var dateOfFeedbackList;
  int? index;
  TeacherFeedback(
      {this.studentid,
      this.feedbackMap,
      this.feedbackList,
      this.dateOfFeedbackList,
      this.index});
  @override
  _TeacherFeedbackState createState() => _TeacherFeedbackState();
}

class _TeacherFeedbackState extends State<TeacherFeedback> {
  final _formkey = GlobalKey<FormState>();
  bool loading = false;
  String? feedback = '';
  @override
  Widget build(BuildContext context) {
    final studentdata = Provider.of<StudentData?>(context);

    var mapfeedback = studentdata?.feedback;

    if (loading == true) {
      return Loading();
```

```
}
return Scaffold(
  backgroundColor: Colors.grey[700],
  appBar: AppBar(
    title: Text(widget.feedbackMap != null ? 'Edit Feedback' : 'Feedback'),
    backgroundColor: Colors.grey[800],
  ),
  body: Column(
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      Form(
        key: _formkey,
        child: Center(
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: ConstrainedBox(
              constraints: BoxConstraints(minHeight: 725.0),
              child: TextFormField(
                style: TextStyle(color: Colors.amber),
                initialValue: widget.feedbackMap != null
                    ? widget.feedbackList[widget.index]
                    : null,
                onChanged: (val) {
                  feedback = val;
                },
                validator: (val) {
                  return val!.isEmpty ? "Please Write a Feedback" : null;
                },
                decoration: textInputDecoration.copyWith(
                    hintText: "Type feedback here"),
                minLines: 35,
                maxLines: 35,
                keyboardType: TextInputType
                    .multiline, // user can press enter to move to the next line
              ),
            ),
          ),
```

```
  ),
),
Padding(
  padding: const EdgeInsets.only(left: 10.0, right: 10.0),
  child: Container(
    width: double.infinity,
    child: TextButton(
        style: ButtonStyle(
          backgroundColor:
              MaterialStateProperty.all<Color>(Colors.amber),
          shape: MaterialStateProperty.all<RoundedRectangleBorder>(
              RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(6.0),
          )),
        ),
        onPressed: () async {
          if (_formkey.currentState!.validate()) {
            setState(() {
              loading = true;
            });
            if (widget.feedbackMap != null) {
              mapfeedback[widget.index] = {
                widget.dateOfFeedbackList[widget.index]: feedback
              };
            } else {
              mapfeedback.add({
                DateFormat('yyyyMMdd').format(DateTime.now()):
                    feedback!
              });
            }
            await DatabaseService(uid: widget.studentid)
                .adddeleteFeedback(mapfeedback);

            setState(() {
              loading = false;
            });
            Navigator.pop(context);
```

```
                }
              },
              child: Text(
                widget.feedbackMap != null
                    ? "Save Changes"
                    : "Send Feedback",
                style: TextStyle(color: Colors.black),
              )),
          ),
        )
      ],
    ),
  );
 }
}
```

EditInputData.dart (*Teacher Screen*)

```
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/teacher/input/InputData.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';

class EditInputData extends StatefulWidget {
  String? studentname;
  String? studentid;
  EditInputData({this.studentname, this.studentid});
  @override
  _EditInputDataState createState() => _EditInputDataState();
}

class _EditInputDataState extends State<EditInputData> {
  bool loading = false;
  @override
  Widget build(BuildContext context) {
    if (widget.studentname == null || widget.studentid == null) {
      return Loading();
    }
    return StreamBuilder<StudentData>(
        stream: DatabaseService(uid: widget.studentid).studentdata,
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            if (snapshot.data!.datainput != null) {
              var dataList = snapshot.data!.datainput;
              var dataInputList = [];
              var dateOfInputList = [];
              for (int x = 0; x < dataList.length; x++) {
                dateOfInputList.add(dataList[x]
                    .keys
                    .toString()
                    .substring(1, dataList[x].keys.toString().length - 1));
```

```
            dataInputList.add(dataList[x]
                .values
                .toString()
                .substring(2, dataList[x].values.toString().length - 2)
                .split(','));
          }

          return loading == true
              ? Loading()
              : Scaffold(
                  backgroundColor: Colors.grey[400],
                  appBar: AppBar(
                    title: Text('${widget.studentname}'),
                    centerTitle: true,
                    backgroundColor: Colors.grey[800],
                  ),
                  body: ListView.builder(
                    itemCount: dataList.length,
                    itemBuilder: (context, index) {
                      return Card(
                        child: Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Column(
                            children: [
                              Row(
                                children: [
                                  Text(
                                    '${DateFormat('dd-MM-yyyy').format(DateTime.parse(dateOfInputList[dataList.length -
index - 1]))}'),
                                  SizedBox(width: 210),
                                  DropdownButton(
                                    dropdownColor: Colors.grey[800],
                                    underline: Container(
                                      color: Colors.transparent,
                                    ),
                                    onChanged: (String? optioninput) {
                                      if (optioninput == 'Edit Data') {
```

```
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => StreamProvider<
                                StudentData?>.value(
                            value: DatabaseService(
                                    uid: widget
                                        .studentid)
                                .studentdata,
                            initialData: null,
                            catchError:
                                (context, error) =>
                                    null,
                            child:
                                TeacherInputStudentData(
                              studentid:
                                  widget.studentid,
                              dataList: dataList,
                              dateOfInputList:
                                  dateOfInputList,
                              dataInputList:
                                  dataInputList,
                              dataInputSelect:
                                  dataInputList[
                                      dataList.length -
                                          index -
                                          1],
                              index:
                                  dataList.length -
                                      index -
                                      1,
                            ))));
              }
              if (optioninput == 'Delete Data') {
                showDialog(
                    context: context,
                    builder: (context) =>
```

```
AlertDialog(
  title: Text("Delete?"),
  content: Text(
      "Data will be permanently deleted."),
  actions: [
    TextButton(
        onPressed:
            () async {
          setState(() {
            loading = true;
          });
          snapshot
              .data!.accList
              .removeAt(dataList
                  .length -
                  index -
                  1);
          snapshot.data!
              .effectList
              .removeAt(dataList
                  .length -
                  index -
                  1);
          snapshot.data!
              .speedList
              .removeAt(dataList
                  .length -
                  index -
                  1);
          snapshot.data!
              .distanceList
              .removeAt(dataList
                  .length -
                  index -
                  1);
          dataList.removeAt(
              dataList.length -
```

```
                                    index -
                                    1);
                            await DatabaseService(
                                    uid: widget
                                        .studentid)
                                .deleteDataInput(
                              snapshot.data!
                                  .accList,
                              snapshot.data!
                                  .effectList,
                              snapshot.data!
                                  .speedList,
                              snapshot.data!
                                  .distanceList,
                              dataList,
                            );
                            setState(() {
                              loading = false;
                            });
                            Navigator.pop(
                                context);
                          },
                          child: Text("Yes")),
                      TextButton(
                          onPressed: () {
                            Navigator.pop(
                                context);
                          },
                          child: Text("No"))
                    ],
                  ));
            }
          },
          items: [
            'Edit Data',
            'Delete Data',
          ].map((String value) {
```

```
          return DropdownMenuItem(
            value: value,
            child: Text(value),
          );
        }).toList(),
        icon: Icon(
          Icons.menu,
        ),
        style: TextStyle(color: Colors.amber),
      ),
    ],
  ),
  SizedBox(
    height: 10.0,
  ),
  Text(
      'No. of Strikes: ${dataInputList[dataList.length - index - 1][0].toString().trim()}'),
  SizedBox(
    height: 5.0,
  ),
  Text(
      'Time in seconds: ${dataInputList[dataList.length - index - 1][1].toString().trim()}'),
  SizedBox(
    height: 5.0,
  ),
  Text(
      'No. of Base Ran: ${dataInputList[dataList.length - index - 1][2].toString().trim()}'),
  SizedBox(
    height: 5.0,
  ),
  Text(
      'Caught Out?: ${dataInputList[dataList.length - index - 1][3].toString().trim()}'),
  SizedBox(
    height: 5.0,
  ),
  Text(
```

```
                                'Coordinates (X,Y): (${dataInputList[dataList.length - index - 1][4].toString().trim()},
${dataInputList[dataList.length - index - 1][5].toString().trim()})'),
                                SizedBox(
                                  height: 5.0,
                                ),
                              ],
                            ),
                          ),
                        );
                      },
                    ),
                  );
            } else {
              return Scaffold(
                backgroundColor: Colors.grey[400],
                appBar: AppBar(
                  title: Text('${widget.studentname}'),
                  centerTitle: true,
                  backgroundColor: Colors.grey[800],
                ),
              );
            }
          } else {
            return Loading();
          }
        });
  }
}
```

EditFeedback.dart (*Teacher Screen*)

```
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/teacher/input/InputFeedback.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';

class EditFeedback extends StatefulWidget {
  String? studentname;
  String? studentid;
  EditFeedback({this.studentname, this.studentid});
  @override
  _EditFeedbackState createState() => _EditFeedbackState();
}

class _EditFeedbackState extends State<EditFeedback> {
  bool loading = false;
  @override
  Widget build(BuildContext context) {
    if (widget.studentname == null || widget.studentid == null) {
      return Loading();
    }
    return StreamBuilder<StudentData>(
        stream: DatabaseService(uid: widget.studentid).studentdata,
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            if (snapshot.data!.feedback != null) {
              var feedbackMap = snapshot.data!.feedback;
              var feedbackList = [];
              var dateOfFeedbackList = [];
              for (int x = 0; x < feedbackMap.length; x++) {
                dateOfFeedbackList.add(feedbackMap[x]
                    .keys
                    .toString()
                    .substring(1, feedbackMap[x].keys.toString().length - 1));
```

```
            feedbackList.add(feedbackMap[x]
                .values
                .toString()
                .substring(1, feedbackMap[x].values.toString().length - 1));
          }
          return loading == true
              ? Loading()
              : Scaffold(
                  backgroundColor: Colors.grey[400],
                  appBar: AppBar(
                    title: Text('${widget.studentname}'),
                    centerTitle: true,
                    backgroundColor: Colors.grey[800],
                  ),
                  body: ListView.builder(
                    itemCount: feedbackList.length,
                    itemBuilder: (context, index) {
                      return Card(
                        child: Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Column(
                            children: [
                              Row(
                                children: [
                                  Text(
                                      '${DateFormat('dd-MM-
yyyy').format(DateTime.parse(dateOfFeedbackList[feedbackList.length - index - 1]))}'),
                                  SizedBox(width: 210),
                                  DropdownButton(
                                    dropdownColor: Colors.grey[800],
                                    underline: Container(
                                      color: Colors.transparent,
                                    ),
                                    onChanged: (String? optioninput) {
                                      if (optioninput == 'Edit Data') {
                                        Navigator.push(
                                            context,
```

```
                MaterialPageRoute(
                    builder: (context) =>
                        StreamProvider<
                            StudentData?>.value(
                          value: DatabaseService(
                                uid: widget
                                    .studentid)
                              .studentdata,
                          initialData: null,
                          catchError: (context,
                                error) =>
                            null,
                          child:
                            TeacherFeedback(
                           studentid: widget
                              .studentid,
                           feedbackMap:
                              feedbackMap,
                           feedbackList:
                              feedbackList,
                           dateOfFeedbackList:
                              dateOfFeedbackList,
                           index: feedbackMap
                                .length -
                              index -
                              1,
                          ),
                        )));
              }
              if (optioninput == 'Delete Data') {
                showDialog(
                    context: context,
                    builder: (context) =>
                        AlertDialog(
                          title: Text("Delete?"),
                          content: Text(
                              "Data will be permanently deleted."),
```

```
                    actions: [
                      TextButton(
                          onPressed:
                              () async {
                            setState(() {
                              loading = true;
                            });
                            feedbackMap.removeAt(
                                feedbackMap
                                        .length -
                                    index -
                                    1);
                            DatabaseService(
                                    uid: widget
                                        .studentid)
                                .adddeleteFeedback(
                                    feedbackMap);
                            setState(() {
                              loading = false;
                            });
                            Navigator.pop(
                                context);
                          },
                          child: Text("Yes")),
                      TextButton(
                          onPressed: () {
                            Navigator.pop(
                                context);
                          },
                          child: Text("No"))
                    ],
                  ));
            }
          },
          items: [
            'Edit Data',
            'Delete Data',
```

```
                            ].map((String value) {
                              return DropdownMenuItem(
                                value: value,
                                child: Text(value),
                              );
                            }).toList(),
                            icon: Icon(
                              Icons.menu,
                            ),
                            style: TextStyle(color: Colors.amber),
                          ),
                        ],
                      ),
                      SizedBox(
                        height: 10.0,
                      ),
                      Text(
                          '${feedbackList[feedbackMap.length - index - 1].trim()}'),
                    ],
                  ),
                ),
              );
            },
          ),
        );
    } else {
      return Scaffold(
        backgroundColor: Colors.grey[400],
        appBar: AppBar(
          title: Text('${widget.studentname}'),
          centerTitle: true,
          backgroundColor: Colors.grey[800],
        ),
      );
    }
  } else {
    return Loading();
```

```
        }
      });
  }
}
```

StudentHome.dart (*Student Screen*)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/student/feedback/ViewFeedback.dart';
import 'package:cstballprogram/services/auth.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:cstballprogram/shared/stats.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

class SHome extends StatelessWidget {
  final AuthService _auth = AuthService();

  @override
  Widget build(BuildContext context) {
    final loginuseruid = Provider.of<cUser?>(context);

    return MultiProvider(
      providers: [
        StreamProvider<StudentData?>.value(
            value: DatabaseService(uid: loginuseruid?.uid).studentdata,
            initialData: null,
            catchError: (context, error) => null),
        StreamProvider<QuerySnapshot?>.value(
          value: DatabaseService().classroomCollectionData,
          initialData: null,
          catchError: (context, error) => null,
        )
      ],
      child: Scaffold(
```

```
drawer: SideBar(),
backgroundColor: Colors.grey[400],
appBar: AppBar(
 title: Text(
  'Swing',
  style: TextStyle(
   color: Colors.amber,
  ),
 ),
 centerTitle: false,
 backgroundColor: Colors.grey[800],
 elevation: 0.0,
 iconTheme: IconThemeData(color: Colors.amber),
 actions: [
  Padding(
   padding: const EdgeInsets.all(8.0),
   child: FloatingActionButton(
    onPressed: () {
     Navigator.push(
       context,
       MaterialPageRoute(
         builder: (context) =>
           StreamProvider<StudentData?>.value(
             value: DatabaseService(uid: loginuseruid?.uid)
               .studentdata,
             initialData: null,
             catchError: (context, error) => null,
             child: StudentFeedbackPage())));
    },
    child: Icon(Icons.remove_red_eye_outlined),
   ),
  ),
  Padding(
   padding: const EdgeInsets.all(8.0),
   child: TextButton.icon(
    style: ButtonStyle(
     backgroundColor:
```

```
                MaterialStateProperty.all<Color>(Colors.amber),
              shape: MaterialStateProperty.all<RoundedRectangleBorder>(
                RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(6.0),
              )),
            ),
            onPressed: () async {
              await _auth.logOut();
            },
            icon: Icon(
              Icons.person,
              color: Colors.grey[800],
            ),
            label: Text('Logout',
                style: GoogleFonts.lato(
                  textStyle:
                      TextStyle(color: Colors.grey[800], fontSize: 15.0),
                )),
          ),
        ),
      ],
    ),
    body: Stats(),
  ),
 );
 }
}

class SideBar extends StatefulWidget {
 @override
 _SideBarState createState() => _SideBarState();
}

class _SideBarState extends State<SideBar> {
 String inputid = '';
 String error = '';
 bool samedata = false;
```

```
bool match = false;
final _formkey = GlobalKey<FormState>();

Future<void> joinClassroom(BuildContext context, var studentclassidlist,
    classroomdatadoc, uid, studentname) async {
  return await showDialog(
      context: context,
      builder: (context) {
        return StatefulBuilder(builder: (context, setState) {
          return Form(
            key: _formkey,
            child: AlertDialog(
              title: Text("Enter Class ID Below"),
              content: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  TextFormField(
                    style: TextStyle(color: Colors.amber),
                    validator: (val) =>
                        val!.isEmpty ? 'Please enter an ID' : null,
                    decoration:
                        textInputDecoration.copyWith(hintText: 'Class ID'),
                    onChanged: (val) {
                      inputid = val;
                    },
                  ),
                  SizedBox(
                    height: 20,
                  ),
                  Text(
                    error,
                    style: TextStyle(color: Colors.red, fontSize: 14.0),
                  )
                ],
              ),
              actions: [
                TextButton(
```

```
        onPressed: () async {
          samedata = false;
          match = false;
          if (_formkey.currentState!.validate()) {
            if (studentclassidlist.length != 0) {
              for (var x in studentclassidlist) {
                if (inputid == x) {
                  setState(() {
                    error = 'Classroom ID already exist';
                    samedata = true;
                  });
                }
              }
            }
            if (samedata == false) {
              if (classroomdatadoc != null) {
                for (var x in classroomdatadoc) {
                  if (inputid == x.get('id')) {
                    await DatabaseService(uid: uid)
                        .addSNewClass(inputid);
                    await DatabaseService(uid: uid, ID: inputid)
                        .addNewStudent(studentname);
                    match = true;
                    Navigator.of(context).pop();
                  }
                  if (match == false) {
                    setState(() {
                      error = 'Classroom ID does not exist';
                    });
                  }
                }
              }
            }
          }
        },
        child: Text("Join"))
    ],
```

```
            ),
          );
        });
      });
}

@override
Widget build(BuildContext context) {
  final student = Provider.of<StudentData?>(context);
  final classroomdataquery = Provider.of<QuerySnapshot?>(context);
  final loginuseruid = Provider.of<cUser?>(context);

  final classroomdatadoc = classroomdataquery?.docs;

  var studentclassidlist = student?.classid == null ? [] : student?.classid;
  var studentclassnamelist = [];

  if (classroomdatadoc != null) {
    for (var x in classroomdatadoc) {
      for (var y = 0; y < studentclassidlist.length; y++) {
        if (x.get('id') == studentclassidlist[y]) {
          studentclassnamelist.add(x.get('classname'));
          break;
        }
      }
    }
  }

  String? studentname = student?.name.toString();
  if (student == null || classroomdataquery == null || loginuseruid == null) {
    return Loading();
  }
  return Theme(
    data: Theme.of(context).copyWith(canvasColor: Colors.grey[400]),
    child: Drawer(
      child: Stack(
        children: [
```

```
Container(
  height: 125.0,
  width: double.infinity,
  child: DrawerHeader(
    decoration: BoxDecoration(
      color: Colors.grey[800],
    ),
    child: Text(
      'Classrooms',
      style: TextStyle(
        color: Colors.amber,
        fontSize: 30.0,
      ),
    ),
  ),
),
Padding(
  padding: EdgeInsets.only(top: 70),
  child: ListView.builder(
      itemCount: studentclassidlist.length,
      itemBuilder: (context, index) {
        return Padding(
          padding: EdgeInsets.only(top: 10),
          child: GestureDetector(
            child: Card(
              margin: EdgeInsets.fromLTRB(20.0, 6.0, 20.0, 0.0),
              child: ListTile(
                  leading: CircleAvatar(
                    radius: 25.0,
                    child: Icon(Icons.sports_baseball),
                  ),
                  title: Text('${studentclassnamelist[index]}')),
            ),
          ),
        );
      }),
),
```

```
          Positioned(
            top: 60,
            right: 20,
            child: Container(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Container(
                    height: 40,
                    width: 40,
                    child: FloatingActionButton(
                      foregroundColor: Colors.grey[800],
                      backgroundColor: Colors.amber,
                      onPressed: () async {
                        await joinClassroom(context, studentclassidlist,
                            classroomdatadoc, loginuseruid.uid, studentname);
                      },
                      child: Icon(
                        Icons.add,
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

ViewFeedback.dart (*Student Screen*)

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';

class StudentFeedbackPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Feedback'),
        backgroundColor: Colors.grey[800],
        elevation: 0,
      ),
      backgroundColor: Colors.grey[400],
      body: StudentFeedbackCard(),
    );
  }
}

class StudentFeedbackCard extends StatefulWidget {
  @override
  _StudentFeedbackCardState createState() => _StudentFeedbackCardState();
}

class _StudentFeedbackCardState extends State<StudentFeedbackCard> {
  @override
  Widget build(BuildContext context) {
    final studentdata = Provider.of<StudentData?>(context);
    if (studentdata == null) {
      return Loading();
    }
    return ListView.builder(
      itemCount: studentdata.feedback.length,
      itemBuilder: (context, int index) {
```

```
      return Card(
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Column(
            children: [
              Text(
                '${DateFormat('dd-MM-
yyyy').format(DateTime.parse(studentdata.feedback[index].keys.toString().substring(1,
studentdata.feedback[index].keys.toString().length - 1)))}'),
              SizedBox(
                height: 5.0,
              ),
              Text(
                '${studentdata.feedback[index].values.toString().substring(1,
studentdata.feedback[index].values.toString().length - 1)}')
            ],
          ),
        ),
      );
    },
  );
 }
}
```

auth.dart (*Services used from Firebase*)

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/services/database.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cstballprogram/screens/authenticate/authenticate.dart';

class AuthService {
  final FirebaseAuth _fAuth = FirebaseAuth.instance;

  // create user obj based on FirebaseUser
  cUser? _userFromFirebaseUser(User? user) {
    return user != null ? cUser(uid: user.uid) : null;
  }

  // auth change user stream
  Stream<cUser?> get loginstatus {
    return _fAuth.authStateChanges().map(_userFromFirebaseUser);
  }

  //sign in with email and pass
  Future signInEmailAndPassword(String email, String password) async {
    try {
      UserCredential result = await _fAuth.signInWithEmailAndPassword(
          email: email.trim(), password: password.trim());
      User user = result.user as User;
      return _userFromFirebaseUser(user);
    } catch (e) {
      print(e.toString());
      return null;
    }
  }

  //register with email & pass
  Future registerEmailAndPassword(
      String name, String email, String password) async {
    try {
      UserCredential result = await _fAuth.createUserWithEmailAndPassword(
```

```
        email: email.trim(), password: password.trim());
    User user = result.user as User;

    //create a new document for the user with the uid
    teacher
        ? await DatabaseService(uid: user.uid)
            .createNewTeacherData(name, [], true, 0)
        : await DatabaseService(uid: user.uid).createNewStudentData(
            name, [], [], [], [], [], 0.0, 0.0, 0.0, 0.0, [], [], false);
    return _userFromFirebaseUser(user);
  } catch (e) {
    print(e.toString());
    return null;
  }
}

//sign out
Future logOut() async {
  try {
    return await _fAuth.signOut();
  } catch (e) {
    print(e.toString());
    return null;
  }
}
}
```

database.dart (*Services used from Firebase*)

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:cstballprogram/models/classroom.dart';
import 'package:cstballprogram/models/user.dart';

class DatabaseService {
  final String? uid;
  final String? ID; //classID
  DatabaseService({this.uid, this.ID});

  late final DocumentReference users =
      FirebaseFirestore.instance.collection('users').doc(uid);

  final CollectionReference collectionUsers =
      FirebaseFirestore.instance.collection('users');

  final CollectionReference classCollection =
      FirebaseFirestore.instance.collection('classrooms');

  //create
  Future createNewStudentData(
    String name,
    var feedback,
    var accuracylist,
    var speedList,
    var effectList,
    var distanceList,
    double accuracy,
    double speed,
    double effect,
    double distance,
    var datainput,
    var classid,
    bool teacher,
  ) async {
    return await users.set({
      'name': name,
```

```
    'feedback': feedback,
    'accList': accuracylist,
    'speedList': speedList,
    'effectList': effectList,
    'distanceList': distanceList,
    'acc': accuracy,
    'speed': speed,
    'effectiveness': effect,
    'distance': distance,
    'datainput': datainput,
    'classid': classid,
    'teacher': teacher,
  });
}

Future updateStudentStatsFromInput(
  var newaccuracy,
  var newspeed,
  var neweffect,
  var newdistance,
  double avgaccuracy,
  double avgspeed,
  double avgeffect,
  double avgdistance,
  var datainput,
) async {
  return await users.update({
    'accList': newaccuracy,
    'speedList': newspeed,
    'effectList': neweffect,
    'distanceList': newdistance,
    'acc': avgaccuracy,
    'speed': avgspeed,
    'effectiveness': avgeffect,
    'distance': avgdistance,
    'datainput': datainput,
  });
```

```
}

Future adddeleteFeedback(var feedback) async {
  return await users.update({'feedback': feedback});
}

//Adds joined classroom's id into student's list
Future addSNewClass(String id) async {
  return await users.update({
    'classid': FieldValue.arrayUnion([id]),
  });
}

//Creates new teacher
Future createNewTeacherData(
    String name, var classid, bool teacher, int numofclass) async {
  return await users.set({
    'name': name,
    'classid': classid,
    'teacher': teacher,
    'numofclass': numofclass
  });
}

//Creates new classrooms
Future createClassroomData(
    String id, var studentid, String classname, String teachername) async {
  return await classCollection.doc(id).set({
    'teachername': teachername,
    'id': id,
    'classname': classname,
    'studentid': studentid,
  });
}

//Adds newly created classroom's id into teacher's list of classroom
Future addTNewClass(String id, int numofclass) async {
```

```
  return await users.update({
    'classid': FieldValue.arrayUnion([id]),
    'numofclass': FieldValue.increment(1),
  });
}

//Add new student into classroom
Future addNewStudent(String? studentname) async {
  return await classCollection.doc(ID).update({
    'studentid': FieldValue.arrayUnion([
      {uid: studentname}
    ])
  });
}

//Delete clasroom's ID from student's list and teacher's list
Future deleteCIDfromUserCList(String classID) async {
  return await users.update({
    'classid': FieldValue.arrayRemove([classID])
  });
}

//Delete classroom's document
Future deleteClassroomDocument(String classID) async {
  return await classCollection.doc(classID).delete();
}

//Delete student from classroom's student list
Future deleteStudentFromClassroomList(
    String studentid, String studentname) async {
  return await classCollection.doc(ID).update({
    'studentid': FieldValue.arrayRemove([
      {studentid: studentname}
    ])
  });
}
```

```
//Delete data input
Future deleteDataInput(var accList, var effectList, var speedList,
    var distanceList, var datainput) async {
  return await users.update({
    'accList': accList,
    'effectList': effectList,
    'speedList': speedList,
    'distanceList': distanceList,
    'datainput': datainput,
  });
}

//teacher data from snapshot
TeacherData _teacherDataFromSnapshot(DocumentSnapshot snapshot) {
  return TeacherData(
    name: snapshot.get('name'),
    classid: snapshot.get('classid'),
    teacher: snapshot.get('teacher'),
    numofclass: snapshot.get('numofclass'),
  );
}

Stream<TeacherData> get teacherdata {
  return users.snapshots().map(_teacherDataFromSnapshot);
}

UserRole? _userRole(DocumentSnapshot snapshot) {
  return UserRole(teacher: snapshot.get('teacher'));
}

Stream<UserRole?> get userrole {
  return users.snapshots().map(_userRole);
}

Stream<QuerySnapshot> get classroomCollectionData {
  return classCollection.snapshots();
}
```

```
ClassDocData _classroomDocumentDataFromSnapshot(DocumentSnapshot snapshot) {
  return ClassDocData(
      classname: snapshot.get('classname'),
      id: snapshot.get('id'),
      studentid: snapshot.get('studentid'),
      teachername: snapshot.get('teachername'));
}

Stream<ClassDocData> get classroomDocumentData {
  return classCollection
      .doc(ID)
      .snapshots()
      .map(_classroomDocumentDataFromSnapshot);
}

StudentData _studentDataFromSnapshot(DocumentSnapshot snapshot) {
  return StudentData(
    name: snapshot.get('name'),
    feedback: snapshot.get('feedback'),
    accList: snapshot.get('accList'),
    effectList: snapshot.get('effectList'),
    speedList: snapshot.get('speedList'),
    distanceList: snapshot.get('distanceList'),
    accuracy: snapshot.get('acc'),
    speed: snapshot.get('speed'),
    effect: snapshot.get('effectiveness'),
    distance: snapshot.get('distance'),
    datainput: snapshot.get('datainput'),
    classid: snapshot.get('classid'),
    teacher: snapshot.get('teacher'),
  );
}

Stream<StudentData> get studentdata {
  return users.snapshots().map(_studentDataFromSnapshot);
}
```

```
}
```

classroom.dart (*Model*)

```
class ClassDocData {
  String? classname;
  String? id;
  var studentid;
  String? teachername;
  ClassDocData({this.classname, this.id, this.studentid, this.teachername});
}
```

datapoints *(Model)*

```
class StatsData {
  StatsData(this.date, this.stat);
  final DateTime date;
  final double stat;
}
```

user.dart (*User Model*)

```
class cUser {
  final String? uid;
  cUser({this.uid});
}

class TeacherData {
  final String? name;
  var classid;
  final bool? teacher;
  final int? numofclass;
  TeacherData({
    this.name,
    this.classid,
    this.teacher,
    this.numofclass,
  });
}
```

```dart
class StudentData {
  final String? name;
  var feedback;
  var accList;
  var effectList;
  var speedList;
  var distanceList;
  final double? accuracy;
  final double? speed;
  final double? effect;
  final double? distance;
  var datainput;
  var classid;
  final bool? teacher;
  StudentData({
    this.name,
    this.feedback,
    this.accList,
    this.effectList,
    this.speedList,
    this.distanceList,
    this.accuracy,
    this.speed,
    this.effect,
    this.distance,
    this.datainput,
    this.classid,
    this.teacher,
  });
}

class UserRole {
  final bool? teacher;
  UserRole({this.teacher});
}
```

constant.dart (*Used in both Teacher and Student Screen*)

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

const textInputDecoration = InputDecoration(
  errorBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Color(0xFF212121), width: 2.0),
  ),
  focusedErrorBorder:
      OutlineInputBorder(borderSide: BorderSide(color: Colors.red)),
  errorStyle: TextStyle(fontSize: 10, height: 1),
  hintStyle: TextStyle(
    color: Color(0xFFBDBDBD),
  ),
  fillColor: Colors.transparent,
  filled: true,
  focusedBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Colors.amber),
  ),
  enabledBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Color(0xFF212121), width: 2.0),
  ),
);

final bases = Container(
  height: 15,
  width: 15,
  color: Color(0xFFBDBDBD),
);

final lato = GoogleFonts.lato(textStyle: TextStyle());
```

loading.dart (*Used in both Teacher and Student Screen*)

```dart
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';

class Loading extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
        color: Colors.grey[800],
        child: Center(
          child: SpinKitCircle(
            color: Colors.amber,
            size: 50.0,
          ),
        ));
  }
}
```

stats.dart (*Used in both Teacher and Student Screen*)

```dart
import 'package:cstballprogram/models/user.dart';
import 'package:cstballprogram/screens/teacher/input/InputData.dart';
import 'package:cstballprogram/shared/constant.dart';
import 'package:cstballprogram/shared/loading.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';
import 'package:cstballprogram/models/datapoints.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

class Stats extends StatefulWidget {
  @override
  _StatsState createState() => _StatsState();
}

class _StatsState extends State<Stats> {
  //The Options User Could Select To Filter Data Based on Different Dates
  final dataperiod = [
    '30-days Average',
    '60-days Average',
    '90-days Average',
    '180-days Average',
    '365-days Average',
    'All-time Average',
  ];

  //The Options User Could Select To View What Specific Visual Data They Would Like To See
  final datadisplayed = ['Ball Location', 'Effectiveness', 'Accuracy', 'Speed'];

  //Initial Selected Data Displayed Is "Ball Location"
  String? selecteddatadisplayed = 'Ball Location';

  //Initial Selected Period Is "30-days Average"
  String? selectedperiod = '30-days Average';

  //Filter Function
```

```
List filter(int selectedperiod, data) {
  var filter = [];
  try {
    DateTime latestdate = DateTime.parse(data[0]
        .keys
        .toString()
        .substring(1, data.last.keys.toString().length - 1));
    //If Selected Period Is Not "All-Time" Then The FOR LOOP Will Proceed To Filter Out The Data
    if (selectedperiod != 1000) {
      for (var z = 0; z < data.length; z++) {
        if (latestdate
                .difference(DateTime.parse(data[z]
                    .keys
                    .toString()
                    .substring(1, data[z].keys.toString().length - 1)))
                .inDays
                .abs() <=
            selectedperiod) {
          filter.add(data[z]);
        }
      }
      return filter;
    } else {
      return data;
    }
  } catch (e) {
    return data;
  }
}

//Averages The Data
double average(data) {
  double total = 0.0;
  int n = 0;
  for (var z = 0; z < data.length; z++) {
    double value = double.parse(data[z]
        .values
```

```
        .toString()
        .substring(1, data[z].values.toString().length - 1));
    //If The Speed Value Is Not "N/A" Which Is Equivalent To Saying "-314", Then "total" Will Be Added As Well As
"n"
    if (value != -314) {
      total += value;
      n++;
    }
  }
  //Returns The Average
  return (total / n);
}

//Converting Data Into A Data Point Model To Be Displayed On The Graph
List<StatsData>? convertToStatData(data) {
  //Contains The Converted Version Of The Data
  List<StatsData> statdatalist = [];

  //Contains A List Of Non-Overlapping Dates From The Data
  List<DateTime> recordeddates = [];

  //FOR LOOP To Loop Through The Dates Of The Data
  for (var x in data) {
    DateTime date = DateTime.parse(
        x.keys.toString().substring(1, x.keys.toString().length - 1));

    //IF Function To Prevent Data Point Inputted On The Same Day To Overlap
    if (!recordeddates.contains(date)) {
      recordeddates.add(date);
      double total = 0.0;
      int n = 0;

      //FOR LOOP To Find Overlapping Dates And Average Them
      for (var z in data) {
        DateTime compare = DateTime.parse(
            z.keys.toString().substring(1, z.keys.toString().length - 1));
        if (date == compare &&
```

```
          double.parse(z.values
                .toString()
                .substring(1, z.values.toString().length - 1)) !=
            -314) {
        total += double.parse(z.values
            .toString()
            .substring(1, z.values.toString().length - 1));
        n++;
      }
    }
    statdatalist
        .add(StatsData(date, double.parse((total / n).toStringAsFixed(1))));
  }
 }
 return statdatalist;
}

@override
Widget build(BuildContext context) {
 final studentdata = Provider.of<StudentData?>(context);
 final datainput = studentdata?.datainput;
 final provacc = studentdata?.accList;
 final provspeed = studentdata?.speedList;
 final provdistance = studentdata?.distanceList;
 final proveffect = studentdata?.effectList;
 var balllocation = [];
 double? avgaccuracy = 0.0;
 double? avgspeed = 0.0;
 double? avgdistance = 0.0;
 double? avgeffect = 0.0;
 var graphacc;
 var graphspeed;
 var grapheffect;

 //Get Data Function To Combine All Functions Above And Prevent Duplication Of Code
 void getData(int days) {
  for (var x in filter(days, datainput)) {
```

```
      balllocation.add(x.values
          .toString()
          .substring(2, x.values.toString().length - 2)
          .split(','));
    }
    avgaccuracy =
        double.parse((average(filter(days, provacc))).toStringAsFixed(2));
    avgspeed =
        double.parse((average(filter(days, provspeed))).toStringAsFixed(1));
    avgdistance = double.parse(
        (average(filter(days, provdistance))).toStringAsFixed(1));
    avgeffect =
        double.parse((average(filter(days, proveffect))).toStringAsFixed(2));
    graphacc = convertToStatData(filter(days, provacc));
    graphspeed = convertToStatData(filter(days, provspeed));
    grapheffect = convertToStatData(filter(days, proveffect));
  }

  if (studentdata == null) {
    return Loading();
  } else {
    if (selectedperiod == '30-days Average') {
      getData(30);
    }
    if (selectedperiod == '60-days Average') {
      getData(60);
    }
    if (selectedperiod == '90-days Average') {
      getData(90);
    }
    if (selectedperiod == '180-days Average') {
      getData(180);
    }
    if (selectedperiod == '365-days Average') {
      getData(365);
    }
    if (selectedperiod == 'All-time Average') {
```

```
    getData(1000);
  }

  return Column(
    children: [
      Container(
        padding: EdgeInsets.all(8.0),
        height: 50,
        width: double.infinity,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            DropdownButton(
              value: selecteddatadisplayed,
              onChanged: (String? value) {
                setState(() {
                  selecteddatadisplayed = value;
                });
              },
              items: datadisplayed.map((String value) {
                return DropdownMenuItem(
                  value: value,
                  child: Text(value),
                );
              }).toList(),
            ),
            SizedBox(width: 40.0),
            DropdownButton(
              value: selectedperiod,
              onChanged: (String? value) {
                setState(() {
                  selectedperiod = value;
                });
              },
              items: dataperiod.map((String value) {
                return DropdownMenuItem(
                  value: value,
```

```
                child: Text(value),
              );
            }).toList(),
          ),
        ],
      )),
    Container(
      padding: EdgeInsets.all(8.0),
      color: Colors.white,
      height: 199.5,
      width: double.infinity,
      child: Row(
        children: [
          Expanded(
            child: Column(
              children: [
                Text(
                  'Accuracy',
                  style: lato.copyWith(fontSize: 25.0),
                ),
                Text(
                  '${double.parse((avgaccuracy! * 100).toStringAsFixed(1))} %',
                  style: lato.copyWith(fontSize: 30.0),
                ),
                SizedBox(
                  height: 40.0,
                ),
                Text(
                  'Speed',
                  style: lato.copyWith(fontSize: 25.0),
                ),
                Text('${avgspeed!} ft/s',
                    style: lato.copyWith(fontSize: 30.0))
              ],
            )),
          Expanded(
            child: Column(
```

```
          children: [
            Text(
              'Effectiveness',
              style: lato.copyWith(fontSize: 25.0),
            ),
            Text(
              '${double.parse((avgeffect! * 100).toStringAsFixed(2))} %',
              style: lato.copyWith(fontSize: 30.0),
            ),
            SizedBox(height: 40.0),
            Text('Distance', style: lato.copyWith(fontSize: 25.0)),
            Text('${avgdistance!} ft',
                style: lato.copyWith(fontSize: 30.0))
          ],
        )),
      ],
    ),
  ),
  Container(
    color: Colors.white,
    child: Divider(
      color: Colors.black,
      height: 3,
      thickness: 3,
      endIndent: 10.0,
      indent: 10.0,
    ),
  ),
  Container(
    color: Colors.white,
    height: 5.0,
  ),
  if (selecteddatadisplayed == 'Ball Location') //T-ball field
    Expanded(
      child: Stack(
        children: [
          Container(
```

```
          color: Colors.green,
        ),
        Padding(
          padding: EdgeInsets.only(right: 45, bottom: 50),
          child: Align(
            alignment: Alignment.bottomRight,
            child: Container(
              height: 475,
              width: 5,
              color: Colors.white,
            ),
          ),
        ),
        Padding(
          padding: EdgeInsets.only(bottom: 50, right: 40),
          child: Align(
            alignment: Alignment.bottomCenter,
            child: Container(
              height: 5,
              width: 375,
              color: Colors.white,
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(right: 45, bottom: 50),
          child: Align(
            alignment: Alignment.bottomRight,
            child: Container(
              height: 240,
              width: 240,
              decoration: BoxDecoration(
                  border: Border.all(color: Colors.white, width: 5)),
              child: Padding(
                padding: const EdgeInsets.all(102.0),
                child: ClipOval(
                  child: Container(
```

```
            color: Colors.brown[600],
          ),
        ),
      ),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.only(bottom: 293, right: 290),
  child: Align(
    alignment: Alignment.bottomRight,
    child: Container(
      width: 100,
      height: 100,
      child: CustomPaint(
        painter: OpenPainter(),
      ),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.only(right: 45, bottom: 275),
  child:
      Align(alignment: Alignment.bottomRight, child: bases),
),
Padding(
  padding: const EdgeInsets.only(right: 270, bottom: 275),
  child:
      Align(alignment: Alignment.bottomRight, child: bases),
),
Padding(
  padding: const EdgeInsets.only(right: 270, bottom: 50),
  child:
      Align(alignment: Alignment.bottomRight, child: bases),
),
for (var index = 0; index < balllocation.length; index++)
  Positioned(
```

```
            child: ClipOval(
              child: Container(
                height: 20,
                width: 20,
                color: int.parse(balllocation[index][2]) == 4
                    ? Colors.amber
                    : int.parse(balllocation[index][2]) == 3
                        ? Colors.cyan
                        : int.parse(balllocation[index][2]) == 2
                            ? Colors.pink
                            : int.parse(balllocation[index][2]) == 1
                                ? Colors.purple
                                : Colors.red,
              ),
            ),
            left: double.parse(balllocation[index][4]),
            top: double.parse(balllocation[index][5]),
          ),
        ],
      ),
    ),
  if (selecteddatadisplayed == 'Effectiveness' ||
      selecteddatadisplayed == 'Accuracy' ||
      selecteddatadisplayed == 'Speed')
    Expanded(
      child: SfCartesianChart(
          primaryXAxis: DateTimeAxis(
              edgeLabelPlacement: EdgeLabelPlacement.shift),
          primaryYAxis: NumericAxis(
              maximum: selecteddatadisplayed == 'Effectiveness' ||
                      selecteddatadisplayed == 'Accuracy'
                  ? 1
                  : null,
              numberFormat:
                  selecteddatadisplayed == 'Effectiveness' ||
                          selecteddatadisplayed == 'Accuracy'
                      ? NumberFormat.percentPattern()
```

```
                        : NumberFormat.decimalPattern()),
                series: <ChartSeries>[
              // Renders line chart
              LineSeries<StatsData, DateTime>(
                  dataSource: selecteddatadisplayed == 'Effectiveness'
                      ? grapheffect
                      : selecteddatadisplayed == 'Accuracy'
                          ? graphacc
                          : graphspeed,
                  xValueMapper: (StatsData stats, _) => stats.date,
                  yValueMapper: (StatsData stats, _) => stats.stat)
            ]))
        ],
      );
    }
  }
}
```