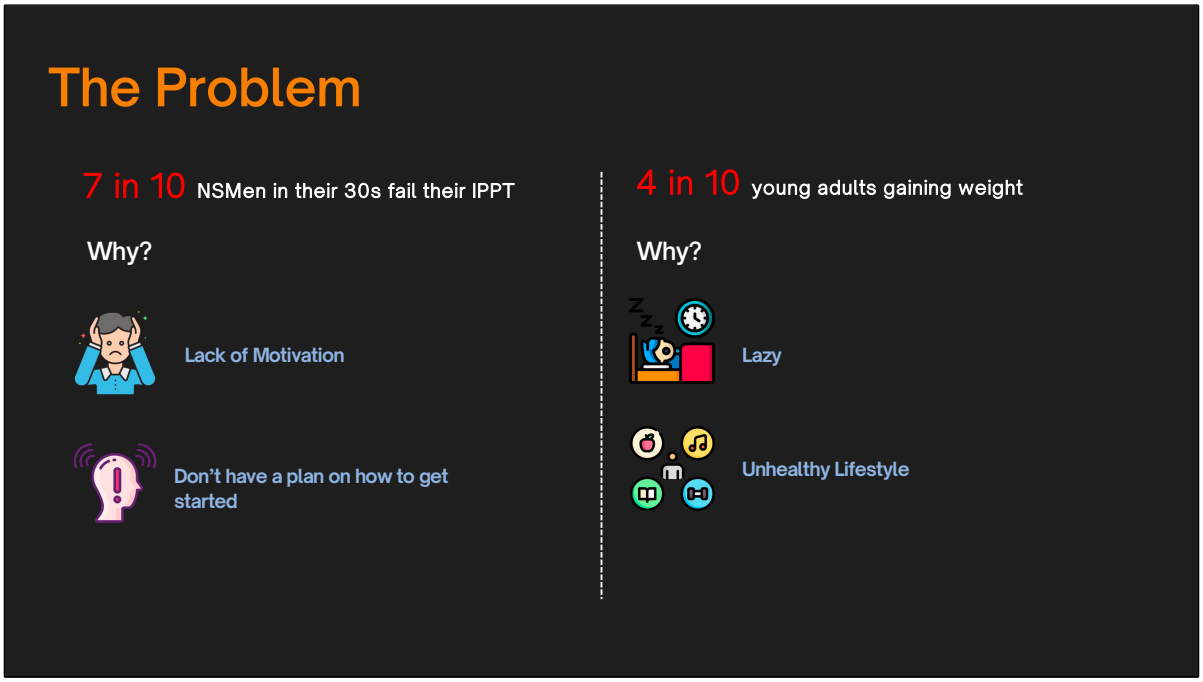
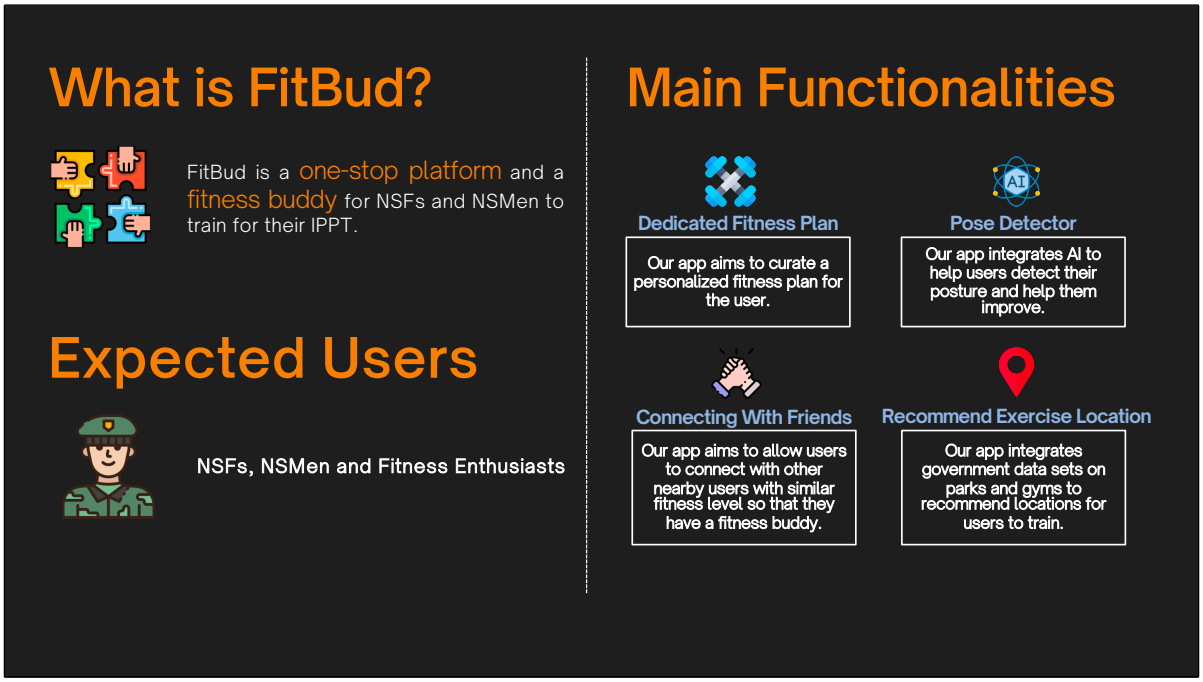
A very good afternoon fellow professor, TA and friends. We are SAMMY and today we'll be bringing to you FitBuds.



Let' start off by understanding the problem at hand. Fitness and health is a concern for many of us. 7 in10 NSMen in 30s are failing their IPPT. Why is this so? This is due to a lack of motivation and the lack of a plan on getting into shape.

In addition, 4 in 10 young adults are gaining weight. This is due to them being increasingly lazy and leading unhealthy lifestyle.

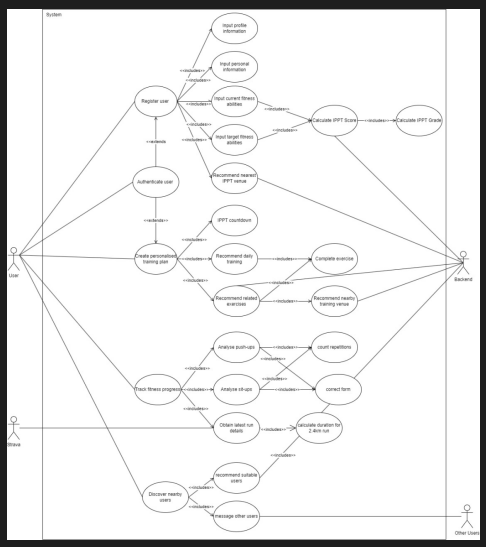
For those who do not know, IPPT is a fitness that residents of Singapore have to undergo as part of their national service obligations. It consists of push-ups, sit-ups and a 2,4km run.



We believe that FitBuds can help alleviate this problem. So what is FitBuds? FitBuds is a one-stop fitness platform and buddy for NSFs and NSMen to train for their IPPT. We expect to onboard NSFs, NSMen and Fitness Enthusiasts.

We have 4 main functionalities. Firstly, a dedicated fitness plan. Our app would be able to curate a personalized training plan for the user based on their target and current fitness levels. Secondly, an AI enabled pose detector that can feedback to the user on their form and posture. This helps a user to maximize their workouts. Thirdly, a social feature that connects users with other nearby users who have similar fitness level. This serves to find a buddy for the user so that they will be grounded and motivated together. Lastly, a recommend exercise location feature that draws upon open-source government database to recommend users on where they can train.

Overview of Use Case Diagram



Main Use Cases

Register User

To prompt for user information during registration.

Authenticate User

To prompt for username and password.

Create Personalized Training Plan

To create a personalized training plan for the user based on their current and target fitness.

Track Fitness Progress

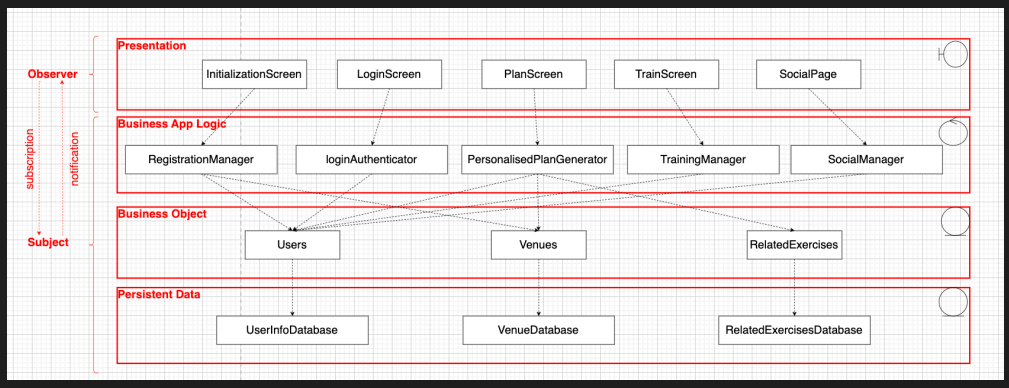
To track user's push-ups, sit-ups and run timings.

Discover Nearby Users

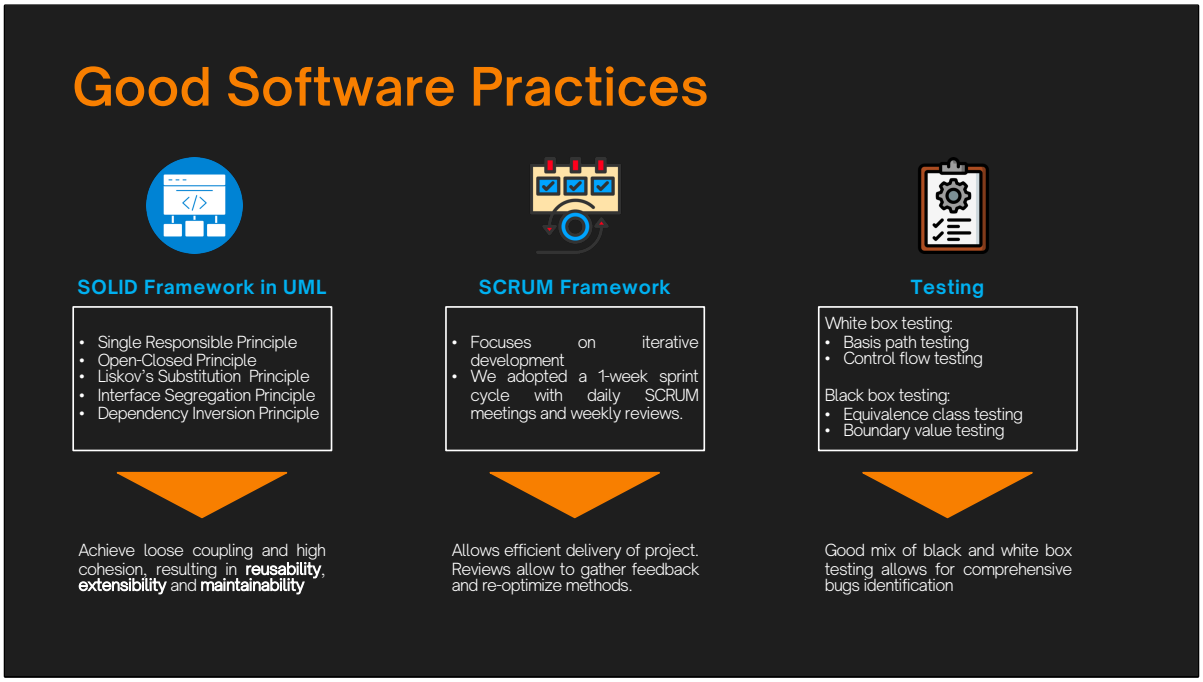
To connect nearby users with similar IPPT scores and arrange meet-ups for users.

As seen, all those features are broken down into various use cases and can be seen in out use case diagram. We have 5 primary use cases: Register User, Authenticate User, Create Personalized Training Plan, Track Fitness Progress and discover nearby users.

Overview of System Design



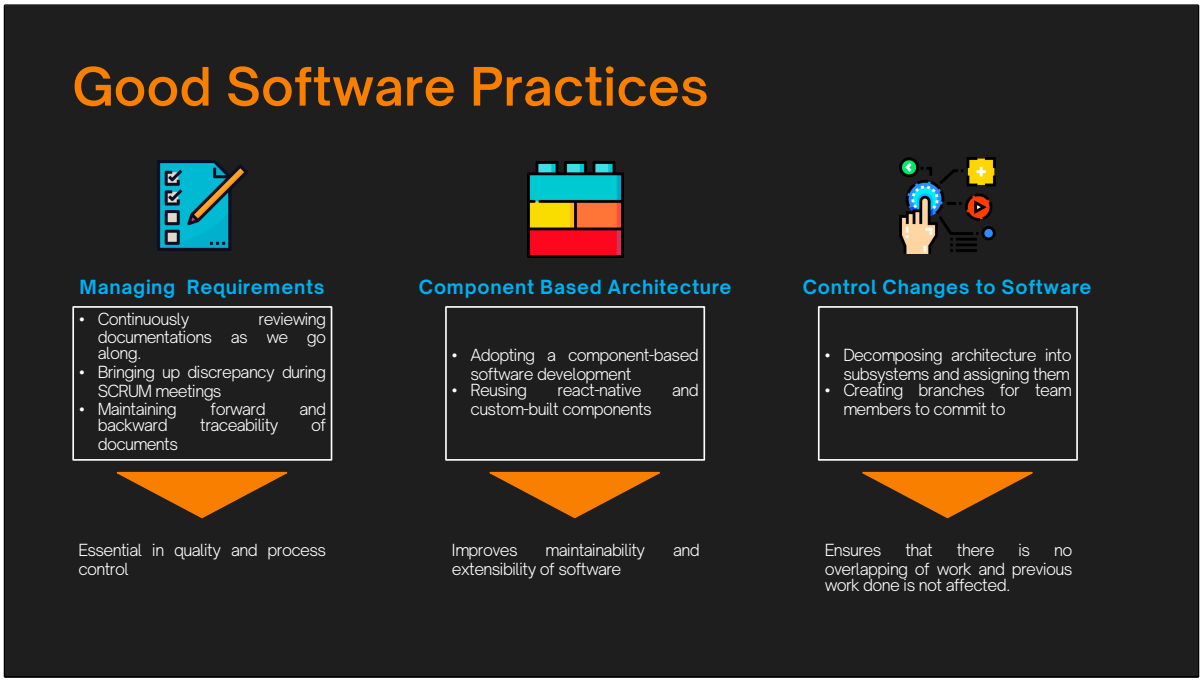
The culmination of our functional requirements, non-functional requirement, design principles, design patterns and constraints leads us to our system architecture



Throughout our entire software engineering process, we have utilized good software engineering processes learnt from our lessons. We utilized the solid framework in our UML modelling. This helps us to ensure that our functions and classes are flexible and easy to maintain and upgrade.

We adopted the SCRUM framework in our development process. We had weekly sprints with daily SCRUM meetings and weekly reviews. This helped us in optimizing our processes which enable an efficient delivery of our project,

Another important aspect of having good software practices is robust testing. We achieve this through a mix of Blackbox and Whitebox testing.



Managing Requirements is essential in quality and process control. We achieved this by continually reviewing our documentation, bringing up discrepancy during SCRUM Meetings and maintaining forward and backwards traceability of documents.

Next, adopting a component-based architecture was another good practice we adopted. We did by reusing react-native and custom-built components. This improves our maintainability and extensibility of our software.

Lastly, it is important to have a mechanism for controlling changes to software. In our case, we decomposed the architecture into subsystems and assigned them to our members. We utilized github in our control mechanism. We created individual branches for our team members to commit to. This ensured that there was no overlapping of current and previous work and that work done is not affected in the event something goes wrong.

How can we support future upgrades?

Having a SOLID framework in our UML reduces dependencies and promotes extensibility. Software can be updated or upgraded without impacting other areas of our code.

Using SCRUM as our methodology allows us to break down our upgrades to smaller pieces. This allows us to build iteratively and efficiently.

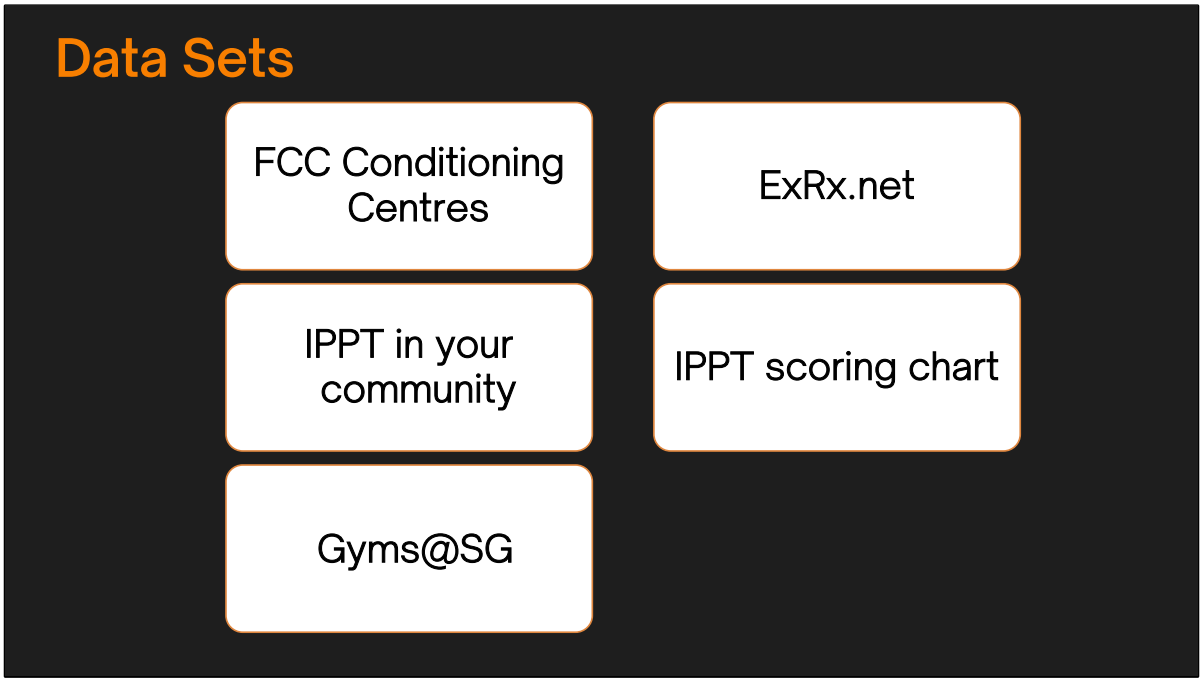
Managing documentation is important because it allows us to find gaps within our software. This aids us in our upgrades.

A component-based architecture allows us to stack components on one another. This makes implementation efficient and simpler.

Good software practices allows us to support future upgrades.

In our case,

Before we dive into our implementation, here is our tech stack. We used React Native for our frontend UI development. TensorFlow for our computer vision model. The backend is implemented and managed using nodeJS, mongoDB and expressJS. Our application runs using cloud which is hosted on AWS.



We have used the following data sets in our app. FCC Conditioning centers, Gyms@SG and IPPT in your community is obtained using Data.gov.sg, a government database. Our exercise database is built using ExRx.net and IPPT Scoring chart is obtained from the internet.

Now we will be moving onto our live demonstration.

Legend: underline refers to the actions to do. Sequence of action.

1. Register new user
    - username: Summit
    - password: ilovesc2006
    - name: Summit Bajaj
    - address: 639798
    - dob: 1 april 2000
    - current: 50, 50, 720
    - target: 60, 60, 600
2. Redirected to plan page
    - Number of days to IPPT
    - Test Location
    - Daily Training
3. Daily training– complete each daily training
4. Click the redirect
5. Click back

6. Bonus exercises
    - Mark them as complete
7. Navigate to social page
    - List of nearby users
    - Click xiao ming to chat function
8. Navigate to profile page
    - User details including age, current grade and ippt test date
9. Navigate to train page
10. Click on 2.4km run
    - Authenticate with Strava
11. Click on pushups
    - Perform pushups (good & bad form)
    - Submit data

Traceability of Input Profile Information

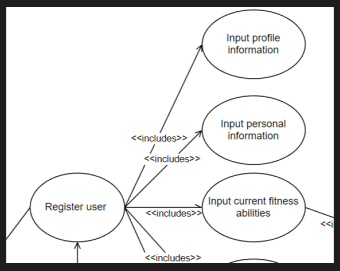
Use Case

Functional Requirement:

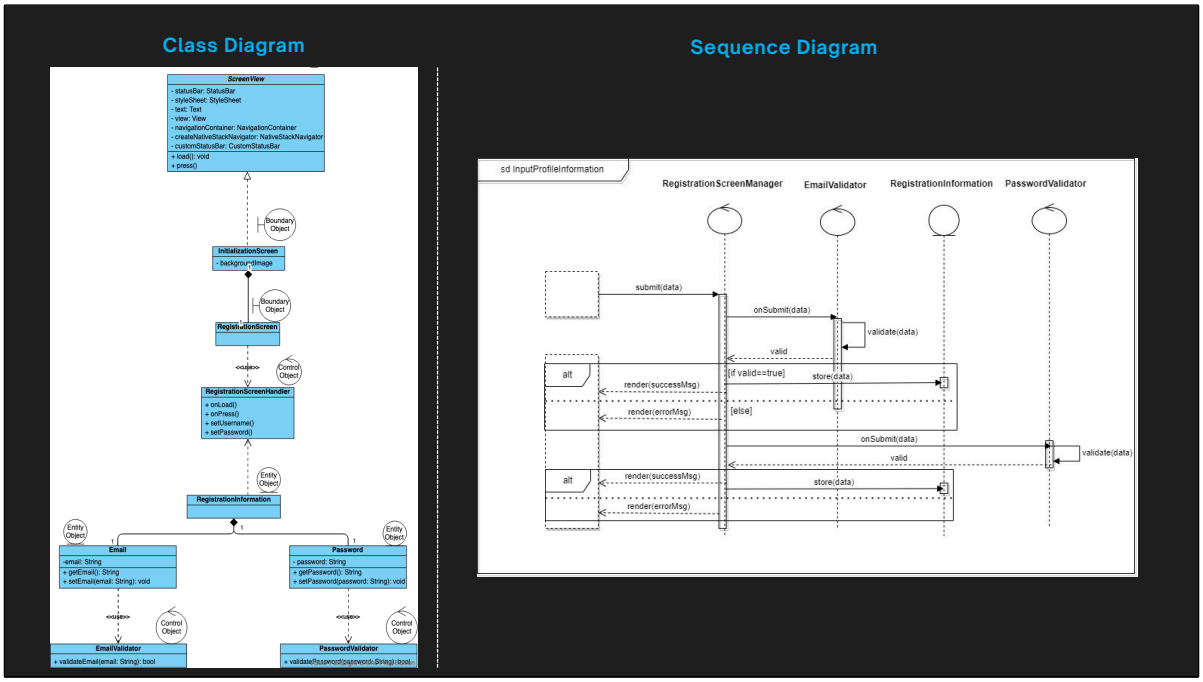
The app must prompt for the user's login information.

1. The app must prompt for the user's username, which must have at least one character.
2. The app must prompt for the user's account password which length must be at least 8, have at least 1 letter, have at least 1 number and must not contain the user's username.

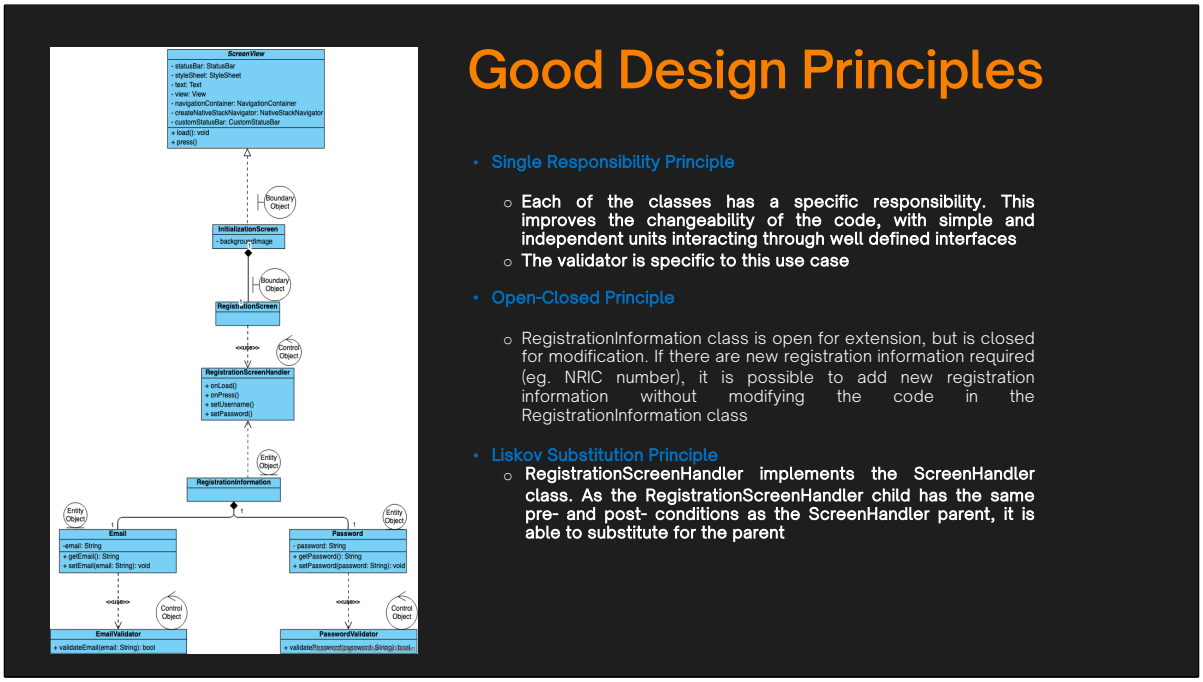
Use Case Diagram



Now that we have seen the demonstration, lets explore the traceability of FitBud's Input Profile Information. The functional requirements are that a user must be able to key in a valid username and password. The corresponding use case diagram can be seen.



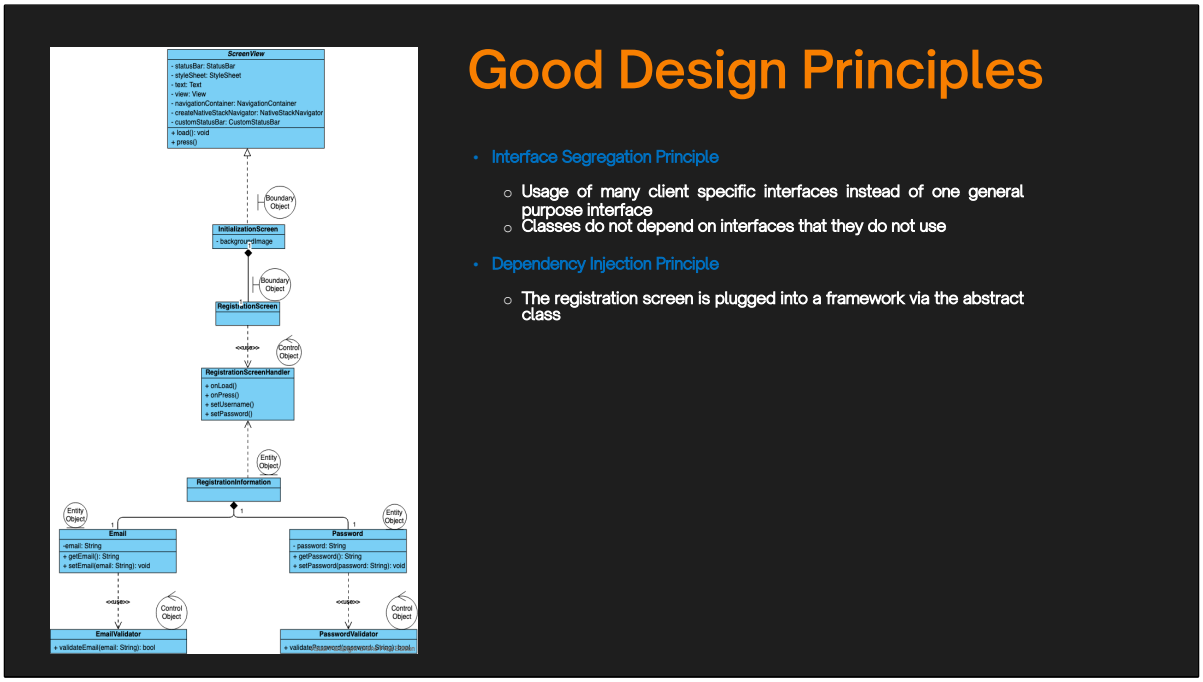
From this, we develop our sequence diagram, as seen. Using that we develop our class diagram. We can see how the use case and sequence diagram come together to form the class diagram with the different boundary, entity and control objects.



We have implemented the solid framework in our use case.

- **Single Responsibility Principle**
    - Each of the classes has a specific responsibility. This improves the changeability of the code, with simple and independent units interacting through well defined interfaces
    - The validator is specific to this use case
- **Open-Closed Principle**
    - RegistrationInformation class is open for extension, but is closed for modification. If there are new registration information required (eg. NRIC number), it is possible to add new registration information without modifying the code in the RegistrationInformation class
- **Liskov Substitution Principle**
    - RegistrationScreenHandler implements the ScreenHandler class. As the

RegistrationScreenHandler child has the same pre- and post- conditions as the ScreenHandler parent, it is able to substitute for the parent



- **Interface Segregation Principle**
  - Usage of many client specific interfaces instead of one general purpose interface
  - Classes do not depend on interfaces that they do not use
- **Dependency Injection Principle**
  - The registration screen is plugged into a framework via the abstract class

Implementation

```
const pwValidator = () => {
  if (password.length > 8) {
    setValidLen(true)
  } else setValidLen(false);

  const regExpChar = /[A-Za-z]/
  if (password.match(regExpChar)) {
    setValidChar(true);
  } else setValidChar(false);

  const regExpDig = /[0-9]/
  if (password.match(regExpDig)){
    setValidNum(true);
  } else setValidNum(false);

  if (email != '' && password.toLowerCase().includes(email.toLowerCase())){
    setValidNoUser(false);
  } else setValidNoUser(true);

  if (validLen && validChar && validNum && validNoUser){
    setValid(true);
  } else setValid(false);
}
```

This is the following implementation in code

Testing

Test Input	Oracle	Log
Password	Expected Output	Actual Output
ilovesc2006	No output message, app allows user to proceed	
johnlovesc2006	"Password contains username", user is not allowed to proceed	

Test Input	Oracle	Log
Password	Expected Output	Actual Output
ilovescse	"Password does not contain any numbers", user is not allowed to proceed.	
12345678	"Password does not contain any characters", user is not allowed to proceed.	

Now that we have seen the requirements, development and implementation. Let's see how it holds through testing. As we see, we do meet our requirements and that inappropriate passwords registers an error.

Traceability of Analyze Push-Up/Sit-Up

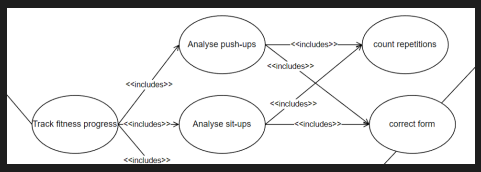
Use Case

Functional Requirement:

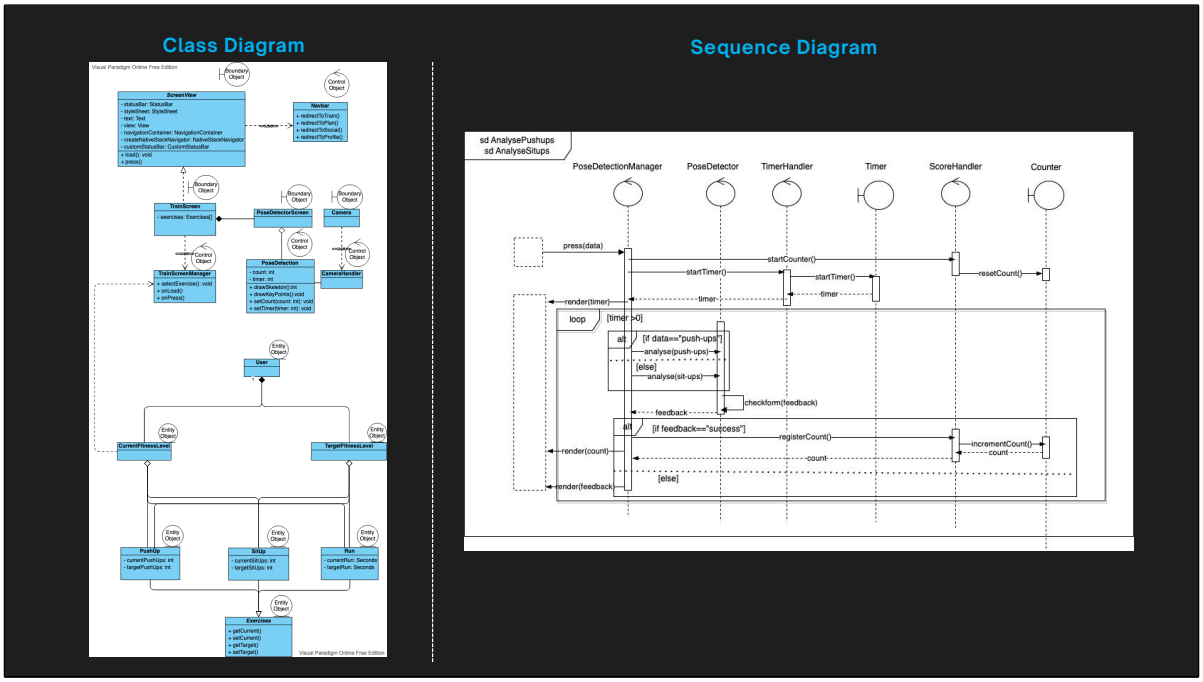
The app must be able to accurately track and correct the user's push-ups during his attempts.

1. The app must allow the user to begin an attempt which will start a 60 second timer.
2. The app must classify the correct push-up form from an incorrect one with an accuracy of 90%.
3. The app must count the number of correct push-ups performed by the user within the attempt.
4. The app must display live feedback on the screen to correct the user's form during the attempt.
5. The app must have the option for users to submit the results of their attempt to be stored.
6. The app must have the option for users to redo their attempt as many times as desired.

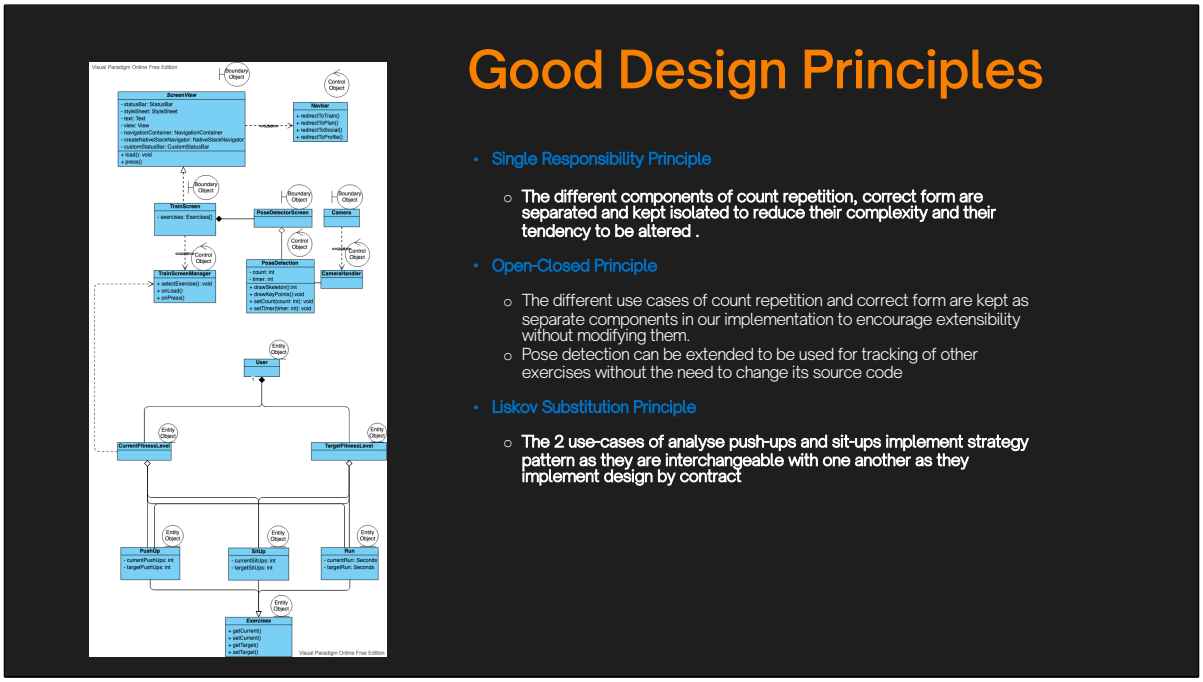
Use Case Diagram



Moving on, we will be looking at the traceability of exciting analyze push-up module. In summary, the requirement is that the app feedbacks the user on the posture of the user's push up and count correct push-ups.



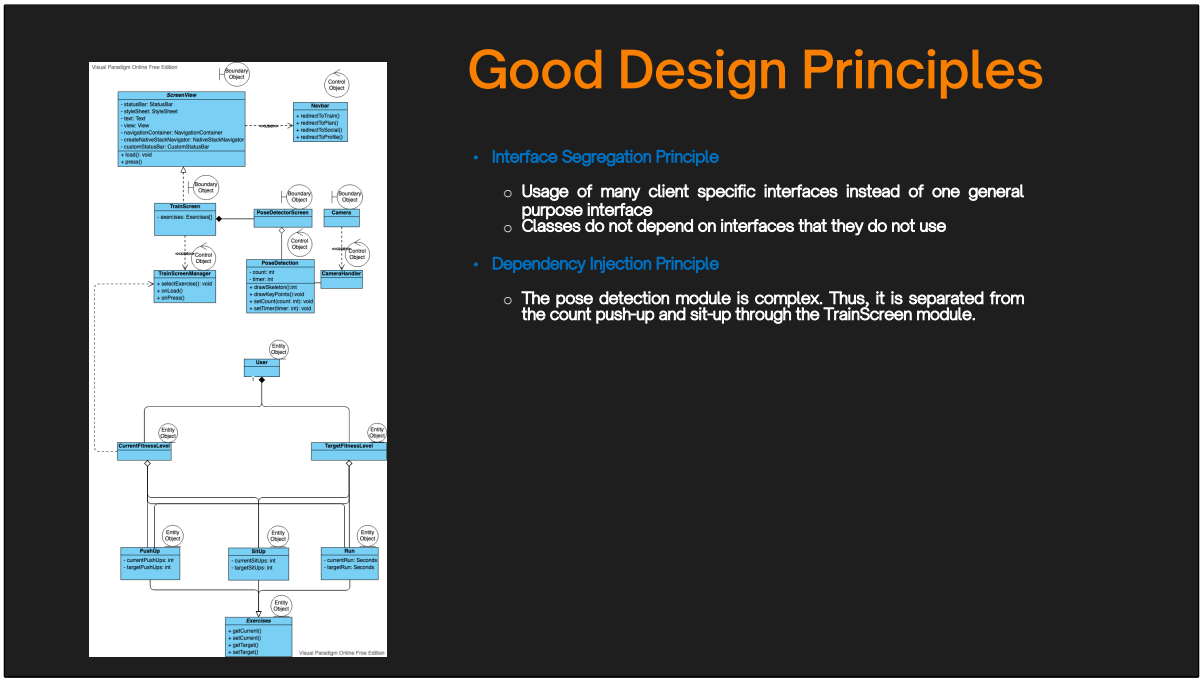
From this, we develop our sequence diagram, as seen. Using that we develop our class diagram. We can see how the use case and sequence diagram come together to form the class diagram with the different boundary, entity and control objects.



Similarly, in this module we have implemented SOLID in our designing of our class diagrams.

- **Single Responsibility Principle**
    - The different components of count repetition, correct form are separated and kept isolated to reduce their complexity and their tendency to be altered .
- **Open-Closed Principle**
    - The different use cases of count repetition and correct form are kept as separate components in our implementation to encourage extensibility without modifying them.
    - Pose detection can be extended to be used for tracking of other exercises without the need to change its source code
- **Liskov Substitution Principle**

- The 2 use-cases of analyse push-ups and sit-ups implement strategy pattern as they are interchangeable with one another as they implement design by contract



- **Interface Segregation Principle**
    - Usage of many client specific interfaces instead of one general purpose interface
    - Classes do not depend on interfaces that they do not use
- **Dependency Injection Principle**
    - The pose detection module is complex. Thus, it is separated from the count push-up and sit-up through the TrainScreen module.

Implementation

Overall Layout

```
<SafeAreaView style={{ flex: 1, justifyContent: "center" }}>
  <TimerDisplay timer={timer} />
  <StartButton setTimer={setTimer} timer={timer} setStart={setStart} />
  {result[0] ? <SubmitButton start={start} score={Math.floor(result[0])} activity={type} /> : null}
  {result[0] ? <ScoreDisplay score={Math.floor(result[0])} /> : null}
  {result[1] ? <DirectionDisplay direction={result[1]} /> : null}
  <Canvas ref={canvasRef} style={{ position: 'absolute', left: 0, top: 0, width: '100%', height: '100%',
  zIndex: 1000, backgroundColor: 'none' }} />
  <ModelCamera model={model} setPredictions={setPredictions} style={{ position: 'absolute', zIndex: 1 }} />
  {result ? <FeedbackDisplay feedback={result[2]} /> : null}
</SafeAreaView>
```

Form correction component:

```
function FeedbackDisplay(props) {
  let itemList = [];
  if (Object.entries(props)[0][1]) {
    for (const [key, value] of Object.entries(Object.entries(props)[0][1])) {
      if (value == false) {
        itemList.push(key + " ")
      }
    }
    return (
      <View style={styles.feedbackContainer}>
        <Text style={styles.feedbackText}>{itemList.length > 0 ? "Incorrect Posture" : "Correct Posture"}:</Text>
        <Text style={styles.feedbackText}>{itemList}</Text>
      </View>
    );
  }
}
```

With that here is the implementation. We can see the various segments of code for the different implementations.

Implementation

Pose detector for push-ups or sit-ups:

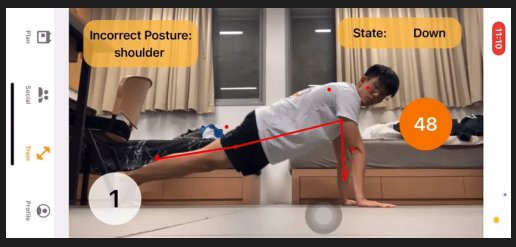
```
start ? setResult(type == "PushUps" ? drawSkeletonPushUps(predictions["keypoints"], 0.5, ctx, 1, 1) :
drawSkeletonSitUps(predictions["keypoints"], 0.1, ctx, 1, 1)) : null;
}
```

Repetition counter component:

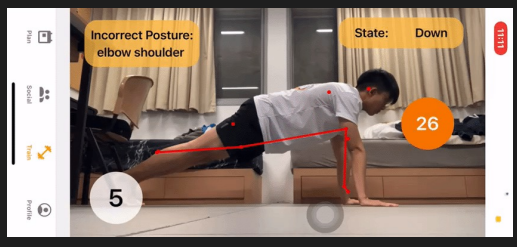
```
function ScoreDisplay({ score }) {
  return (
    <View style={styles.scoreContainer}>
      <Text style={styles.scoreText} >
        {score}
      </Text>
    </View>
  );
}
```

Testing

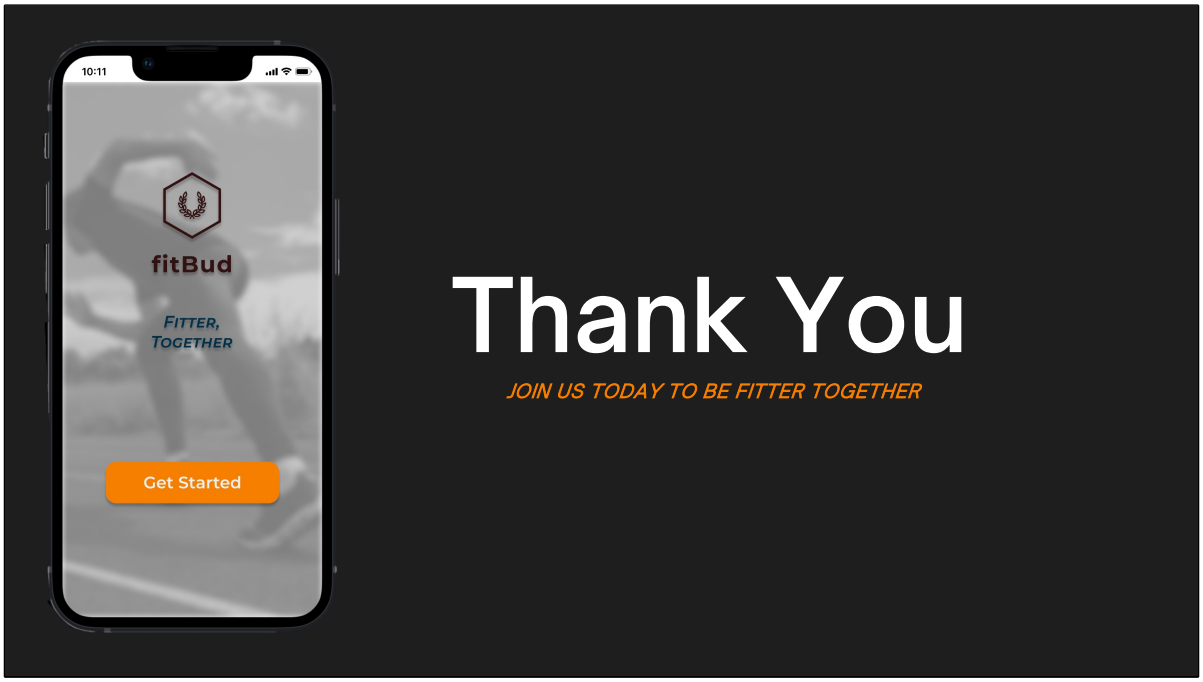
Successful Push-up Attempt:



Unsuccessful Push-up Attempt:



With that, we can see how it holds up with the testing. The app is able to count a successful push up and provide feedback for an unsuccessful one.



With that we believe that people can get fitter with FitBud with our features. We hope that you join FitBud today to be fitter. Thank you.