smart Chair

Mackenson Jean, *EE*

Annavay Kean, *CpE*

Thien Nguyen, *CpE*

Bonarine Ramjas, *EE*

Senior Design I

Spring 2019

University of central florida

[**1.** **Executive Summary** 1](#_Toc5894463)

[**2.** **Project overview** 2](#_Toc5894464)

[2.1. Background 2](#_Toc5894465)

[2.2. Group dynamics 2](#_Toc5894466)

[2.2.1. Roles & Responsibilities 2](#_Toc5894467)

[2.3. Specifications & requirements 3](#_Toc5894468)

[2.3.1. Software 3](#_Toc5894469)

[2.3.2. Hardware 4](#_Toc5894470)

[2.3.3. Disclaimer 5](#_Toc5894471)

[2.4. Milestones 5](#_Toc5894472)

[2.5. Constraints 6](#_Toc5894473)

[**3.** **Research Related to project** 8](#_Toc5894474)

[3.1. Spine/posture health 8](#_Toc5894475)

[3.1.1. Skeletal 8](#_Toc5894476)

[3.1.2. Muscular 8](#_Toc5894477)

[3.1.3. Circulation 8](#_Toc5894478)

[3.1.4. Center or Gravity 8](#_Toc5894479)

[3.1.5. Recommended Exercises 9](#_Toc5894480)

[3.2 Sensing User Presence 11](#_Toc5894481)

[3.2.1 Sensing Posture 12](#_Toc5894482)

[3.3 Similar products 12](#_Toc5894483)

[3.3.1 Upright GO 12](#_Toc5894484)

[3.3.1 Wearable device (non-smart) 12](#_Toc5894485)

[3.3.2 ergonomic chairs 13](#_Toc5894486)

[**Table 3.1:** Competition Benefit Table 13](#_Toc5894487)

[3.4 Hardware 13](#_Toc5894488)

[3.4.1 Power Supply 13](#_Toc5894489)

[3.4.2 Load sensors 16](#_Toc5894490)

[3.4.3 Proximity sensors 17](#_Toc5894491)

[3.4.4 Amplification circuit 17](#_Toc5894492)

[3.4.5 Microcontrollers 17](#_Toc5894493)

[3.4.5.1 Communication Protocols 18](#_Toc5894494)

[3.4.6 Haptic Motor 18](#_Toc5894495)

[3.4.6.1 LRA 18](#_Toc5894496)

[3.4.6.2 ERM 18](#_Toc5894497)

[3.4.7 Batteries 19](#_Toc5894498)

[3.4.8 Hardware selection 20](#_Toc5894499)

[3.4.8.2 Piezoresistive Sensor 25](#_Toc5894500)

[3.4.8.3 Development board 27](#_Toc5894501)

[3.4.8.4 PCB 28](#_Toc5894502)

[3.4.8.4.1 Design software 28](#_Toc5894503)

[3.4.9 Schematic Overview 29](#_Toc5894504)

[3.5 Software 29](#_Toc5894505)

[3.5.1 Security 30](#_Toc5894506)

[3.5.2 Framework 32](#_Toc5894507)

[**3.5.3** **Important Ideas of the Dart Language** 40](#_Toc5894508)

[3.5.4 User Data 41](#_Toc5894509)

[3.5.4.1 Local Storage 41](#_Toc5894510)

[3.5.4.2 External Storage 44](#_Toc5894511)

[3.5.4.3 Database 45](#_Toc5894512)

[3.5.4.4 SQL 46](#_Toc5894513)

[3.5.5 Database-Application Interfacing 51](#_Toc5894514)

[3.5.6 **Firebase** 53](#_Toc5894515)

[**3.5.7** **Bluetooth** 56](#_Toc5894516)

[3.5.8 Password Recovery 59](#_Toc5894517)

[**4** **Standards & Constraint** 61](#_Toc5894518)

[**4.1** **Distribution - App Store Requirements (to be a subsection in Standards)** 61](#_Toc5894519)

[**4.2** **Chair electricity Standards** 63](#_Toc5894520)

[**4.3** **Power Supply Standards** 63](#_Toc5894521)

[**5** **Initial Design** 70](#_Toc5894522)

[5.1 Software 70](#_Toc5894523)

[5.1.1 Application and Database 70](#_Toc5894524)

[5.1.1.1 User Interface 70](#_Toc5894525)

[5.1.1.1.1 General Design 71](#_Toc5894526)

[5.1.1.1.2 Widgets 71](#_Toc5894527)

[5.1.1.2 Database Design with MySQL 72](#_Toc5894528)

[**5.1.1.2.1** **Database Design with Firebase** 73](#_Toc5894529)

[5.1.2 Embedded Software 74](#_Toc5894530)

[5.2 Hardware 74](#_Toc5894531)

[5.2.1 Bluetooth Module 74](#_Toc5894532)

[5.2.2 Bluetooth 2.0 EDR Modules 74](#_Toc5894533)

[5.2.3 HC-05 74](#_Toc5894534)

[5.2.4 HC-6 75](#_Toc5894535)

[5.2.5 Bluetooth 4 / BLE Modules 75](#_Toc5894536)

[5.2.6 HM-10C 76](#_Toc5894537)

[5.2.7 HM-10S 76](#_Toc5894538)

[5.2.8 BLE Link Bee 76](#_Toc5894539)

[5.2.9 Implemented Bluetooth Module 76](#_Toc5894540)

[5.2.10 Bluetooth Breadboard Experiment 79](#_Toc5894541)

[5.2.11 PCB 81](#_Toc5894542)

[5.2.12 Proximity Sensor 81](#_Toc5894543)

[5.3 Milestones 82](#_Toc5894544)

[**6** **Appendix** 84](#_Toc5894545)

[**6.1** Works Cited 84](#_Toc5894546)

[7.2 Permissions Requested / Acquired 86](#_Toc5894547)

# **Executive Summary**

The project would be a smart chair to encourage healthy habits in an office job. One of the many issues facing office workers is back pain caused by long periods of time sitting coupled with poor posture. The product would include a ‘pad’ that would be placed onto an office chair and would gather and interpret data on the user’s sitting trends. The smart chair would monitor how long the user sits and remind them to get up at a regular interval. The device would also monitor the user’s posture and provide feedback. The smart chair would pair with an application on the user’s phone to provide feedback and reminders. The feedback will come from research related to common spine issues and simple techniques to provide relief. It would be beneficial for all people and offices because the smart chair will be powered by a low voltage power load and rechargeable. The battery itself will power all sensors, LEDs and Bluetooth transceiver. Due to the back pain resulting from sitting too long in a chair, the smart chair will vibrate after a predetermined or user defined interval, and it will tell you how long you should get up and stretch or walk for. The data being used to advise and correct the user’s habits will primarily be collected using a grid of load or pressure based sensors. By using these, the design could be considered more “plug and play” for existing office chairs.

# **Project overview**

# Background

The National Institute of Neurological Disorders and Stroke claims that back pain is the most common job-related disability. Around 80% of adults will experience back pain during their lifetime and 25% of adults have experienced back pain recently.

According to Spine-health’s website, sitting in office chairs involves a ‘static posture’ that can cause or worsen back pain. The increase in back pain is caused by over-stretching the spinal ligaments due to posture used when sitting in an office chair (slouching). A prolonged slouch posture can cause damage to the spine. It is advisable to maintain proper posture when sitting in an office chair to reduce the cause of this pain. Ergonomic chairs can assist in improving back support, but will not stop back pain unless the proper posture is used.

According to Spine-health, proper posture includes making sure your office chair is adjusted appropriately as well as your sitting posture.

How to determine if the office chair is adjusted correctly:

* User’s elbows should be able to reach the keyboard and mouse while maintaining a 90-degree angle with the spine.
* Ensure that there is not too much pressure on the lower thigh by fitting two fingers between the seat and the user’s thighs. If there is too much pressure on the thigh, correct with a footrest or raise the chair height.
* Check the chair back by ensuring a fist can fit between the end of the chair and the user’s calf. If there is not sufficient room, adjust the back of the chair.
* The user’s back should be pressed to the back of the chair such that the lower back support built-in to the chair pushes the lower back slightly outward. This will protect against back pain and will encourage better posture.
* User’s eye level should be level with the center of the monitor. Adjust monitor as necessary.
* Arm rests should be high enough to slightly lift user’s shoulders. This will reduce pressure on spine.

Regular standing breaks can aid in reducing back pain induced by office chairs. Spine-health recommends getting up from an office chair at least once every half hour for at least one to two minutes.

# Group dynamics

# Roles & Responsibilities

**Annavay:**

Annavay will be the lead for the user interface design due to her prior experience in the area. She will work with Thien on the application’s backend to write the algorithm to interpret the incoming sensor data. She will also work with Thien to ensure proper communication between the application and the hardware.

**Thien**:

Thien will work along with Annavay on the user interface and backend algorithm to take in information from user and allowing them to customize a set amount for a timer and notification. Also help the integration process between software and hardware to communicate with the Bluetooth to synchronize with the application, allowing it to notify users and set guidelines.

**Bonarine**:

Bonarine will take point on the sensor & peripheral (LED,bluetooth module)  integration during the design and test phases. He will support Mackenson with PCB design, using it as an opportunity to learn eagle, and share responsibilities as stated below.

**Mackenson:**

Mackenson will take charge to design the PCB and make sure we have the perfect circuit that we need for the project, and work with Bonarine to implement the power that will be used for the sensor, Bluetooth and led systems. As a group, we will work together to integrate the software and hardware while testing proper functionality.

# Specifications & requirements



**Figure 2.1:** High level Project overview

# Software

* The application should allow the user to log in.
* The application should save user data.
* The application should protect user privacy.
* The application should allow the user to set the time interval to be reminded to get up.
* The application should have default settings according to scientific evidence available.
* Default time interval between walking reminders should be 30 minutes.
* The application should remind the user to get up after specified time interval.
* The application should gather user posture data.
* The application should provide feedback on the user’s posture.
* The application should remind the user to get up after specified sitting interval.



**Figure 2.2:**  First Draft User Interface

2.2.1.1 Agile

<https://www.agilealliance.org/agile101/>

In order to maximize efficiency during the development and ensure that the project requirements are met, the developers are interested in using Agile development.

Agile is a model for sustainable development cycles. Agile advocates for the use of ‘sprints’ which serve as chunks of time devoted to specific feature development. After the customer and the project manager negotiate requirements, the developers plan out a series of sprints. Each sprint will concentrate on developing one section of the application. No other requirements can be added to development once a sprint has begun. Each sprint should last no more than one month. During a sprint, there should be a fifteen minute tag-up among developers where each member must respond to the following questions: What did I accomplish yesterday? What am I going to work on today? Only under very rare circumstances should a sprint be canceled.

After a sprint has ended, the development team should have an extended meeting (usually a few hours). During this meeting, the members should discuss how the last sprint went. Did something not get completed? Did something go particularly well? After discussing this, it is necessary to plan the next sprint. At this time, requirements can be added or updated.

# Hardware

* The device should update the sensor input readings every 30 seconds.
* The device should pair with a phone app and computer software.
* 5 day battery life, minimum.
* PCB not to exceed 230 cm squared.
* Sensor array should be able to load a minimum of 100lbs (45.4kg).
* Vibrate module should operate between 200-400 Hz or rpm equivalent.
* LED notification lights should not exceed 1000 lumens (or equivalent rating)



**Figure 2.3:** First Draft Hardware layout

# Disclaimer

The intent of this project is to study common issues associated with poor sitting habits including slouching, leaning, crossing, legs, and prolonged periodsand through various data gathering techniques, offer suggestive actions to the user that could serve to reduce or mitigate, the previously referenced issues as well as offer data trends for the . It is not intended to permanently heal or remedy more severe issues such as herniated discs, nerve damage/pain, poor circulation due to preexisting conditions, or issues that require ongoing treatment from a medical professional.

# Milestones

**Hardware**

* + 1. Circuit Design & software simulation including calculations and filter/gain stages
    2. breadboard simulation with test parts
    3. Design PCB & simulate
    4. Order final parts
    5. Order PCBs
    6. Solder the parts on the PCB and reflow
    7. Test PCB

**Software**

1. Storyboard the user interface
2. Design Database
3. Learn how to create a mobile application for android and learn how to use local storage.
4. Create Database and populate with sample data.
5. Create skeleton UI and get interaction with backend.
6. Algorithm to analyze the data from hardware and to recommend posture adjustments.
7. Communication with hardware to send commands to LEDs and receive sensor data.
8. Finished version of UI.
9. Fully functional application backend.

**Integration**

1. Send and receive data via Bluetooth
2. User setting controls LED’s
3. Read sensor data from PCB’s

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Primary** | **Secondary** | **Status** |
| Bluetooth integration | Thien |  | proto |
| Database development | Annavay | Thien | proto |
| microcontroller | Rudy | Mackenson | ordered |
| power supply | Mackenson | Rudy | proto |
| rechargeable battery | Mackenson | Rudy |  |
| force sensor integration | Rudy | Mackenson | proto |
| vibration module | Rudy |  | on order |
| user interface | Annavay | Thien |  |
| PCB Design | Mackenson | Rudy | 1st iteration |

**Table x: Tasks & Responsibility**

# Constraints

This serves to illustrate the constraints imposed upon our design based upon decisions we will make in subsequent sections relating to software and hardware.

**House of Quality**

This house of quality serves as guide to our technical and customer focused requirements. The result of which is coincides with a defined specification that will ultimately serve as a benefit or limitation to our design.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Technical Requirements** | Battery Life | PCB size | Sensor Load | Vibration module frequency | Sensor reading | LED brightness | Bluetooth interface between chair and mobile application. | Push notifications | Database for user data. | Simple application interface |  |
| **Customer Requirements** |  | ˅ | ˅ | ˅ | ˄ | ˄ | ˅ | ˄ | ˄ | ˅ | ˄ |  |
| Receive feedback on current posture | ˄ |  |  |  | + | + | + |  | + |  |  |  |
| Chair must maintain mobility similar to unmodified chair | ˅ | - | - |  |  |  |  |  |  |  |  |  |
| Comfort of chair must be similar to unmodified chair | ˅ | - | - | - | - |  | - |  |  |  |  |  |
| The chair should pair with a mobile application. | ˄ |  |  |  |  | - |  | + | + | + |  |  |
| The application should provide meaningful feedback. | ˄ |  |  |  | + | + |  | + | + | + | + |  |
| The application should keep a history of user data. | ˅ |  |  |  |  |  |  | - |  | + | + |  |
| The application should be simple to use. | ˄ | - |  |  | - | - |  |  |  |  | + |  |
|  | **Specifications** | ≥5 days | ≤230cm^2 | 100lbs | 200-400Hz | every 30 sec | ≤1000 lumens |  | 5 minutes ( for demo) |  | ≤10 clicks for a task |  |

**Table 2.1:** HoQ

# **Research Related to project**

# Spine/posture health

Below we will discuss some of the motivation and research behind the need for our device, and some of the issues we can potentially solve. In a subsequent section, we will discuss the suggestions we intend to offer the user to help mitigate some of the issues that arise from prolonged sitting and poor seated posture.

# Skeletal

The vertebra column (the human spine) can be grouped into 5 regions: cervical, thoracic, lumbar, sacrum, and coccyx, however the last two begin the pelvis, tailbone, and peripheral structures of the body’s lower region. The spine as a whole system is compromised of vertebrae which house the central nervous system, interconnected by sponge like material known as a disc. Our project focuses on the midthoracic down to coccyx region of the spine as this is where most issues arise from sitting with poor posture for long periods of time. It should be noted that with proper posture, long periods of sitting can cause fatigue in similar regions of the body. The lumbar region of the spine, is by far the most common area affected by poor posture. Focusing on the lumbar spine, the region typically follows a *lordotic curve,* or more simply, a backwards “*C*” and deviations in this shape are where discomfort begins; note that from person to person this curvature naturally favors a more flat or more arched shape.

# Muscular

In addition to the skeletal, there is also the muscular anatomy of the back. While much of the focus traditionally goes to the spine with regards to back health, the muscles play a far more important role and tend to be where many of us feel discomfort.

# Circulation

An issue that is often overlooked with respect to sitting for extended periods of time is circulation. Sitting requires less effort than most activities which leads a person to usually being at resting heart rate; often associated with lower blood pressure. This coupled with several muscle groups holding fixed positions, and the constriction of blood vessels in certain regions, specifically those supplying lower extremities, can lead to reduced circulation within the body. The results of poor circulation vary from discomfort, such as a the tingling of a limb “falling asleep”, to more severe lifelong issues.

# 3.1.4. Center or Gravity

<https://www.verywellhealth.com/human-center-of-gravity-296568>

An integral part of this project is learning how to interpret sensor data from the chair in order to determine how the user is sitting. One idea is to use the grid-like sensor data to determine the user’s center of gravity. In a peer-reviewed article published by Very Well Health, they discussed a study where it was determined that lower-back pain correlated to a center of gravity further back than average. This study also concluded that individuals with lower-back pain would have strength and balance challenges when trying to improve posture.

# 3.1.5. Recommended Exercises

<https://www.verywellhealth.com/yoga-for-back-pain-supine-spinal-twist-297346>

<https://www.doyouyoga.com/5-yoga-poses-to-ease-lower-back-pain-30237/>

<https://www.yogajournal.com/poses/sphinx-pose>

<https://www.yogaoutlet.com/guides/how-to-do-thread-the-needle-pose-in-yoga>

<https://www.yogaoutlet.com/guides/how-to-do-thread-the-needle-pose-in-yoga>

According to DoYouYoga.com, there are yoga poses to help alleviate lower back pain. Below we will discuss some of these poses in order to aid in the alleviation process.

**Surpine Twist**

It is important to note that this type of pose can cause herniated disks if not performed correctly. If performed correctly, it can help with back pain.

How to do:

1. Lay on your back.
2. Move your right knee over to the left side of your body. Hold.
3. Return to lying flat on your back.
4. Move your left knee over to the right side of your body. Hold.

**Sphinx Pose**

How to do:

1. Lay on your stomach.
2. Move your elbows under your shoulders, keeping your forearms parallel to the floor.
3. Press upward in a curve. Hold.

**Thread the Needle**

This pose is for all experience levels. The aim of the pose is to stretch the shoulders. This pose can help with neck, back, and shoulder pain. Do not attempt this pose if knees, shoulders, or neck is [injured](https://www.yogaoutlet.com/guides/how-to-do-thread-the-needle-pose-in-yoga).

How to do:

1. Begin on hands and knees with wrists beneath shoulders and knees beneath hips.
2. Slide right arm under left arm. Right palm should be facing up.
3. Hold for 1 minute. Repeat on other side.

**Cat and Cow Pose**

This pose is a general-purpose stretch that can help with spine [flexibility](https://www.yogaoutlet.com/guides/how-to-do-cat-cow-pose-in-yoga). It can help to correct spine alignment with persistent use. Pregnant users should not perform the cat pose, only the cow pose.

How to do:

1. Begin on hands and knees with wrists beneath shoulders and knees beneath hips.
2. Cow Pose: Drop stomach toward ground and lift chest and chin toward ceiling.
3. Cat Pose: Draw back up and round back up toward ceiling.
4. Repeat 5-20 times.

**Double V Pose**

[**https://blog.paleohacks.com/yoga-release-shoulder-back-pain/**](https://blog.paleohacks.com/yoga-release-shoulder-back-pain/)

How to do:

1. Begin on stomach supporting weight on forearms.
2. Adjust right forearm such that fingers are pointing left. Adjust left forearm such that fingers are pointing right.
3. Slowly move hands in the direction the fingers are pointing. Rest head on yoga block.
4. Hold, then switch sides by moving the rear forearm to the front. Repeat.

**Child’s Pose**

<https://blog.paleohacks.com/yoga-release-shoulder-back-pain/>

How to do:

1. Begin on hands and knees.
2. Move big toes toward each other and move knees outward. Rest hips on heels.
3. Move hands straight out and lower chest. Let forehead rest on mat. Hold.

**Arm Across Chest Pose**

[**https://www.openfit.com/9-yoga-poses-to-help-relieve-neck-and-shoulder-pain**](https://www.openfit.com/9-yoga-poses-to-help-relieve-neck-and-shoulder-pain)

How to do:

1. Sit straight up on a chair or on the floor. The neck should be elongated and the shoulders should be relaxed.
2. Stretch out the right arm and cross it over your chest such that it is parallel with the ground.
3. Turn your head to look over your right shoulder. Hold, then switch sides.

**Eagle Arms Pose**

[**https://www.openfit.com/9-yoga-poses-to-help-relieve-neck-and-shoulder-pain**](https://www.openfit.com/9-yoga-poses-to-help-relieve-neck-and-shoulder-pain)

How to do:

1. This pose can be done sitting or standing. Elongate the neck and straighten the back.
2. Stretch arms out such that they are parallel to the ground and perpendicular to the chest.
3. Bend the right arm at the elbow such that the fingers on the right hand point skyward.
4. Wrap the left hand under and around the right arm such that the palms of the hands touch. Hold, then switch sides.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pose** | **Lower Back Pain** | **Upper Back Pain** | **Neck / Shoulder Pain** |
| Cat - Cow | **✔** | **✔** | **✔** |
| Thread the Needle |  | **✔** | **✔** |
| Supine Twist | **✔** | **✔** |  |
| Sphinx Pose | **✔** |  |  |
| Double V Pose |  | **✔** | **✔** |
| Child’s Pose | **✔** |  |  |
| Arm Across Chest Pose |  |  | **✔** |
| Eagle Arms Pose |  | **✔** | **✔** |

**Table X: Summary of yoga poses for back pain.**

# Sensing User Presence

A key aspect of the design’s functionality is knowing when the chair is occupied by its user. This relates to several hardware and software components. If the chair is not occupied we would not want the Bluetooth module enabled and transmitting data, we would also like the sensors and power supply to disable or enter low power functionality (if applicable). If the user leaves their chair It is also crucial to know if the chair is occupied someone other than the user to whom the devices profile is paired with, this results in the prevention of providing the user incorrect data, as well maintaining security of the device by not transmitting sensitive user data, which could result in a breach of privacy. In the event that this product were to enter a commercial market, many of these requirements would apply to standards relating to energy efficiency, which are listed in the respective section of this paper. In order to sense the user’s seating habits and posture we begin with the true or false, is the user present? This will be determined with a proximity sensor located in the backrest of the chair and the ten piezo sensors located throughout the base and back of the chair. There are a several methods to accomplish this. One involves using an interrupt associated with the state of the proximity sensor and some number of the piezo sensors. Once the combination is triggered the interrupt flag can be raised, activating the other sensors, begin to take readings and establish a connection with the user’s cellphone via Bluetooth link. From there, another interrupt can be set to trigger after the sensors do not receive data for a predetermined amount of time which can reset the flag for the first interrupt and trigger a low power/off option for the device.

# Sensing Posture

One of the more complicated parts of this project will be deciding how to determine if the user is sitting with proper posture. One idea is based on measuring the user’s center of gravity as discussed earlier in this document. Since the sensors will be in a grid layout, it would be possible to calculate the user’s center of gravity. Recalling from the general research, that a center of gravity further back indicates lower back pain. So theoretically, the device could determine if the user is susceptible to lower back pain, and use that fact to recommend stretches or yoga poses to alleviate pain. If the chair can detect this user’s center of gravity, it could also provide a reminder for the user to improve how they are sitting.

# Similar products

According to a report done by market research firm *MarketsandMarkets*, the IoT health devices market has growth projections of more than tripling by 2023, from approximately twenty billion dollars (USD) in 2018. This market includes various wearable technologies, measuring apparatus’, and other connected devices designed to digitize the consumer health market. In the following sections we will discuss some products that overlap in marketspace while discuss some of their pitfalls which we hope to improve upon.

# 3.3.1 Upright GO

The upright GO is a body contact adhesive device that primarily marketed towards users with poor standing or walking posture. This focus does not take into account users who sit for prolonged periods of time. The device is attached to the users upper back, along the thoracic region of the spine, specifically, between the shoulder blades. This suggests the device focuses on slouching or forward leaning bias’. Priced at approximately eighty dollars plus tax and shipping, at the writing of this design brief. With an average rating across amazon and the OEM website of four out five stars, this device fills the need of its market. The direct contact with the user and the necessity of continually purchasing adhesive strips in order to wear the device may dissuade potential users of such device. Our device aims to provide the user with a plug and play solution, once the chair is calibrated to its user, the only maintenance required is charging the device.

# Wearable device (non-smart)

Another tried and true method for correcting posture is the use of tension straps. These devices are capable of targeting upper, lower, and combination areas of the back. The principle is simple, a strap holds the problem areas in a fixed position, using tension to prevent the user from returning to a poor posture. The straps that target slouching typically do so by pulling back the shoulders, this is because a square shoulder position prevents rounding of the back. These devices range in cost from tens to thousands of dollars, based on several factors including target region(s) and how custom it is to the user (note that some of these devices come with medical recommendations from medical professionals). Something to consider with a device like this is that since the strap is holding the tension, it may take the user longer to develop the muscular strength required to hold the body in the optimal position, if at all. The device also covers a relatively large portion of the body, as such if worn under clothing it would require periodic washing or even replacing, and if worn over clothing may cause the user discomfort or even awkwardness. Referring to custom medical devices, these typically come in the form of braces. These devices tend to be bulkier and the primary driver of cost is custom molding to the user. These devices are designed with the intent of having the user wear them long term as the amount of support they provide reduces the likelihood of muscular development required to maintain a healthy posture long term. These devices can serve a workforce demographic that sits long term, as we defined for our project, but the bulk and cost of these devices justifies our project’s purpose and cost.

# ergonomic chairs

The initial inspiration for this project is derived from the structure of a traditional office chair. While most office chairs offer some level of ergonomic comfort, the primary driver of posture correcting chairs is cost.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Upright Go | Strap/harness | Ergonomic chair |
| Cost | + | 0 | - |
| Overall Effectiveness(user response based) | + | + | + |
| Lower back | - | + | 0 |
| Upper back | + | + | 0 |
| Lower extremities | - | 0 | - |
|  |  |  |  |

# **Table 3.1:** Competition Benefit Table

# Hardware

This section serves to summarize relevant technologies and considerations with respect to the various hardware and software to be used in our design. The topics covered are not final decisions but necessary research and decision matrixes weighing requirements of our design.

# Power Supply

The goal of this project is to develop a sophisticated smart chair that can save people health problem, yet easy to use smart office chair that will make professor’s life and people working in office better. With the health problem that can kill people, the idea of smart chair as a senior design project will use technology to save life and make people more active. In the world of technology, power supply places an important role in the product that will use power to function. In addition to power supply, we should make a decision to one of the best and standards that will use in our project so that we can use the national and international standards where our products can sell around the world. Our main objective is to have safety in our supplies because using electrical components can cause severe damages. Choosing a power supply that will meet the international standard is the best possibility for us because we are using smart chair that can save people’s live. The power supply will need to have the proper conductor and insulator to protect the smart chair against electric shock and making sure it is safe to move around because the people who are going to use the chair will not seat in one place, so it is important to use a power supply that meets all the safety requirements to save live because it will have a battery power supply implemented in it. As you know, batteries are the hazardous. It is important to use the standards rule to regulate the chair because people are going to seat on it. Batteries can cause severe burns and can be exploded when it is full of charge and being used under a volume of weights. The power supply that we will use in this project will meet the American, European standards to prevent the electric shock such as, fire, explosions, burns, hot wires. We will have to do all the calculations necessaries to pick the power supply cords and the adapter systems that will use to convert form AC to DC meet the requirements (standards) to prevent against all the shock because smart chair will not be able to only use in the United states and European but around the world. As we know, every project should meet the national and international standards, below are some standards that use around the world for any projects that contain power supply specifically explosive stuff like batteries that can kill and cause some negative effects on humans being and materials.

Back in the days precisely in the 20th century, electricity was a major problem in the world of technology. Thanks to some scientists in the 21st century that used their brain to beat the challenge by inventing electricity DC to AC and AC to DC, now we can use technology in our products with a low cost. The smart chair will use a power AC source of 120V and 60 Hz for the North America which designates that the alternative current alternates 60 times per second in the substation where it comes from. As our project which is Smart Chair will have to use DC current, we will need a wall adapter that convert AC to DC to run power to our board. All the equipment in the smart chair will use 5 volt direct current but the wall has 120V, we will need to stepdown the voltage that come from the wall then using a rectifier and a voltage regulator that will keep the components running at a constant voltage so that the components will not burn and explode due to unstable voltage rate.

First, the AC power in the wall is 120V. we are using a stepdown transformer from 120V to 5V because the most of our components are using 4,5 to 5V to electrify the smart chair project. These calculations can run our systems that are using low power material to run all the equipment that needs power to function. As you know the input voltage is not always perfect, we will need a capacitor that connected in parallel to the load, regulator and a diode will be used to regulates the DC voltage from the transformation AC to DC so that the battery that will use in the system can charge automatically and keep the charge for an amount of time because the smart chair will be connected to the wall every single minutes. To do so the battery will be charged and be using for about one week of charge that enough for a smart office chair which is not going outdoor for long amount of times.

According to Wikipedia using, a rectifier in the project will help us to take the alternating current AC power in a different way and transform it to Direct current (DC) to reach the amount of current that we are using on the load. As we learned in electronics classes, we have two (2) types of rectifier: half-wave rectifier and full-wave rectifier. The half wave can use in a single phase like our projects, but it will have a negative effects on the smart chair because it will ignore the negative sinusoidal ways to being zero that we have in the input as a step function we will have one half of the input waveform that will reach the output waveform which means it will have a lower voltage. Compare to the half-wave rectifier, the full wave rectifier will convert the full input waveform to one with polarity constant. The advantages of using the full wave rectifier are more effective and convert both polarities of the input to direct current DC and it produces the voltage higher than the half-wave, and it has a positive effect on the negative sinusoidal input where it creates constantly a continuous power supply on the circuit. Due to the advantages of the full-wave on the circuit, we will use it on our project to stabilize the direct current DC that we will have in the smart chair where people will use often to seat and doing their work. We will have to accomplish the change from the alternate current to direct current that will need the main important electrical supply to achieve the function of a full wave rectifier that we have to calculate and build the design by using the diodes, capacitor and resistor to connect to the sensor where the full project can achieve.

Using a full-wave diode connection can help us have a better project because it will have 4 diodes connection to get the transformation that we need. One of the diode functions will help to get the direct current in the right direction and have two pairs of diodes that will conduct the direct current DC in one direction. Having four diodes in the rectifier will be divided in two halfwave, one half wave for the positive direction and the other halfwave for the negative direction of the cycle that will help us to get the positive cycles to go the right direction that we need to power the load in direct current DC. Through to the full wave, we will have to add a capacitor that should place in parallel with the load to regulate the current going to the load which is the straight line that we need to have in the direct current DC. The reason that we need the capacitor is to charge and discharge the alternate current AC input when they are switching one to the other by a sinusoidal wave to correct from sine wave to line that we need to get in direct current for the transition from one peak to the other peak. The capacitor will store charge as case of a rupture where it will provide backup voltage to the load and prevent the components without any loss of power.

**Power usage**

Moving the chair everywhere is the most important aspect that will have to pick a power supply to achieve the longest distances from the wall that can reach the AC source, a rechargeable battery-operated source will be employed for the last long possible charge that can power the chair for the distance. A lithium-Ion 9 Volt battery and 1200 mAH will be using in this project to power all the features materiel that are essential to monitor the chair and preserve it from discharge. It will persist functional at the very minimum of one-week life that can bring the chair everywhere with a full guarantee of charge because we are using the low power mode system on the chair. When the chair is not in movement the low power mode function is used to save energy and battery life. The battery and a power supply of 120V AC input power to connect to the wall to charge the battery. The power supply will have 9V DC output to charge the battery that will need the direct current as output.

As we are using a rechargeable battery, it will allow the chair’s owner replaces the battery in case it has something wrong without the assistance of a professional that will charge them a lot of money, they will be able to do maintenance on the battery with a free cost. However, the chair by itself will have several dedicated technical services on the software that will allow the customers to see all the information related to the battery and the components inside the smart office chair. The software will indicate the battery’s life, and when they have to do a quit maintenance on the chair like the battery position and when to change the battery to make the world technology easier and applicable for all situation. The battery will have some commands to the board and send signal when it is full of charge and discharge, then it will be beeping, and the red LED will be lighting to allow the user that it time to charge. As long as the battery is full the green LED will be light up. One of our purpose in this project will be using the printing circuit board to distribute the power in each electrical component. A 9V battery will be too much to power the PCB because it requires a 5V to run and 3 to 4V to the rest of the component. We will be using a step-down autotransformer with a combination of a rectifier to find the amount of voltages that will allow to power the PCB and the electrical component to get the accurate quantity of power at an unchanging rate. Selecting a 9V lithium battery will give the amount of voltage that each component needs at low cost.

# Load sensors

In order to obtain consistent and reliable data about the user’s seating habits an array of sensors must be utilized. The composition of this sensor array will be determined based on weighing cost, efficiency, accuracy, and integration. Below, we will explore the different types of sensors that would have applications in our design as well as draw comparisons between each to conclude the ideal fit, based on the previously stated parameters. The beam style load cell, in its simplest iteration, is comprised of a deformation element, such a spring, a strain gauge, and some form of housing these elements. As the element deforms the strain gauge will also deform, thus resulting in a change in the strain gauge’s resistivity. This results in a signal on the order of a few to tens of millivolts. The element paired with the strain gauge determines the range and longevity of the sensor, these are typically larger devices when compared to sensors such as piezoresistive force sensors. The strain gauge offers longevity of the sensor and useful in applications where the load can be static for prolonged periods. The load cell can be subcategorized as one of the following: pneumatic, hydraulic, or electric. For the purposes of this discussion we will be referring to electric options.

The device converts a compressive (or tensive) input signal to a variable voltage output. Taking a look at the load cell technology, the sensor is comprised of a point of contact that is attached to a sealed cylinder filled with a conductive matter, with some output wires, housed in a sealed container. This container is usually cylindrical. The material inside the housing will undergo a change in electrical property which is read as a voltage or current change. From this change, typically a wheatstone bridge circuit will sense a change in resistance, providing an output signal, which can be processed and interpreted. Sensor calibration can be achieved with calibration weights and the specific sensor’s datasheet, detailing the corresponding output. These cells are very similar to the strain gauge where a change in electrical property is induced under load. The key distinction is that here, a voltage is induced, as is characteristic of piezo-resistivity. This works well in situations where the load continually varies. In applications where the load is static, i.e. it is applied and does not change over a an undefined period, we find there won’t be an accurate output.

# Proximity sensors

Sensing the surrounding area for objects with respect to a reference point is the purpose of a proximity sensor. These will typically emit some range of electromagnetic radiation, and look for either changes in the returning field, or the return field itself. The type of proximity sensor required is based on the object(s) being targeted. In the following section we will discuss common proximity sensors and the basic theory behind each. From this we will conclude the ideal choice for our design. Sensing the distance between the user’s upper back and their chair back requires a sensor that will be subject to being covered for long periods of time (when the user is actively in the chair and seated in an appropriate posture). Similar to ultrasonic sensors, microwave sensors detect proximity/motion of an object using waves. The key distinctions are in the spectrum of operation and the speed and accuracy improvements due to the use of higher frequencies, over ultrasonic sensors. They come with their own drawbacks which include higher susceptibility to interference, which is due to their spectrum of operation being the same as a large number of wireless communication devices and methods.

# Amplification circuit

With the use of many of the aforementioned sensor technologies, their outputs register on scales as small as one-one hundredth of the input. One such sensor would be the compressive load cells that operate on a few volts and register changes electrical changes in the range of millivolts or tenths or ohms. In order to turn these outputs into useable data we must magnify these minute changes. This is where the use of various gain circuits comes in. We may consider a simple op-amp circuit where we can calculate the necessary gain required to supply our MCU with a reliable, interpretable signal. Op-amps offer low cost gains and simple calculations to achieve consistent results.

# Microcontrollers

At the core of most digital electronic devices, is a module that interfaces peripheral input, processes it and provides it to the end user in a meaningful way. For many of these types of devices, this module is a microcontroller. By standard design, a microcontroller consists of some onboard memory, a central processing unit (CPU), and I/O pin(s) which can be general purpose. These devices are often considered self-contained systems, or specialized computers based on their applications. The versatility of an MCU, makes it an ideal choice for our design, in comparison to an FPGA or actual computer. With respect to programmability, there is a wide range of languages and development platforms. These will be discussed in another section. Microcontrollers can increase in complexity based on the number of GPIO’s, operating supplies, the addition of other sub systems on the integrated-circuit (IC), such as analog-to digital converters (ADC’s,), AC-DC converter IC’s, and communication protocols.

# Communication Protocols

As mentioned before, communication protocols are a crucial aspect of transmitting data, as a subsystem of the microcontroller. This determines what and how data can be transmitted. I2C, bus, UART and several other communication protocols offer unique capabilities over another as we will discuss below. We keep in mind a few of the considerations being made for our project which include the ability to receive data from multiple inputs while providing data to multiple outputs in near real time, the timing intervals required for these sequences. Also, we should consider the types of data in and out such as the ability to support ADC.

# Haptic Motor

A crucial aspect of our posture correcting measures is notifying the user of poor seating positions, and interval-based reminders to stretch. These would offer a less distracting method of communicating with the user as compared to a flashing light or sound. This is countered by the fact that the vibration could startle the user, resulting in consequences such as breaks of concentration, discomfort, and in some cases, severe health issues. It should be noted that if done in a manner similar to the haptic drivers of cellphones, these consequences could be minimized. Below we will explore the common types of haptic sensors used in cellular devices.

# LRA

LRA’s rely on an AC voltage to drive a voice coil that is in contact with a spring-loaded mass. Vibration is achieved when the resonant frequency of the spring is produced by the voice coil. This offers fewer moving parts which improves the longevity of the part however, in small applications where DC is the preferred power supply a transformer is required. This also allows for more precise and varied patterns of vibration, but intensity is limited by the resonant characteristics.

# ERM

As the name suggests, the ERM relies on offset mass that rotates. This typically operates on a DC supply. They are comprised of a small motor, an eccentric mass, and a shaft connecting the two. It is akin to a washing machine that isn’t properly balanced, the result is the machine shaking, or vibrating. The feedback is varied by the input, providing for more range and intensity of the driver, however because we rely on moving parts, the precision and patterns that can be produced is limited compared to LRA’s. these motors are far more common that LRA’s, provide better market access and costing.

# 

# Batteries

**Lithium-ion**

To achieve this project, we need backup power for smart chair can be more dependable and help people working in the office more active. As Smart can be indoor and outdoor, we decided to do it smart by using all equipment necessary to make it comfortable for people and used a 9V battery which is not heavy for the chair and can charge for one week without connected to the wall. This is enormously important to find a better way for our professor and people working in office to have a smart chair that will tell them when to have some exercises and taking a walk. We use 9V lithium battery because it will have adequate voltage to be able to use on our apparatuses proficiently, it will be last longer because it will be using a full charge for at least a week**,** it can replace easily in case the maintenance is needed, and it can be in the board without taking a risk of unprotected battery.

The implement of a 9V battery will be good enough to power the chair. Since our system will run with 4 to 5V, we will need to decrease the power dissolute of the battery by using voltage divider circuits analysis to find how we will use the 9V battery in each of the component where we can choose the perfect stability between proficiency and size. When we pick this project, we already think about how to make the office chair smart for people and can be used with backup power which is a battery. The battery will work in the main circuit connected to board and will share the positive end with the load, and we connected negative to the ground side where it will be protected by the battery safety and standards. The diode will be connected to the end of the positive side to the battery that will come in action when the electrical component is not getting power. As in the project we accompanied a rectifier to the in the input of the PCB to the key load, the full-wave diode will be used as open circuit in contradiction of all the contrary side of the current that will be in the project.

Having a Bluetooth to connect to cellphone and computer to display information about the chair, different number of sensors for the load, we will use LED that will allow the customer when to stand up and doing exercises as we will have more items that will consume power, we are planning to have the power usage that will be around 70 watts which is not going to be bad for the smart chair. All the components will be using 5V from the battery which are involved energy that we need to control because we want the smart chair to use less energy as possible well-organized and precise to power the systems with all electrical components that we need to make this project works. We will use the American standards in our project where we are using the power from the industry that we have in the wall which is 120V and 60 HZ frequency that we used in United States of America. We will use the wall power to recharge the chair when the battery is about to discharge, and we will add a battery to deliver power to the chair when the adapter is not connected, and the chair is outdoor. The smart chair project will be able to use outside the US specially in Europe where they use 220V and 50 Hertz, in that case we will need another adapter that can connect to the European system as the input and give the same amount of voltages as the output to provide the chair. At that time, we do not need to redesign the PCB where we use the international safety in our project.

# Hardware selection

Below, we will discuss the various products being considered for our design and ultimate decisions as well as justifications. This section will also be the location of design revisions and changes with respect to components as with any design, though it is not wanted, changes can occur.

* + - 1. Microcontroller

The selection of microcontroller is the hardest choice to do in the project because it is the brains that will control all parts in our projects. The most important thing that the microcontroller does as it acts as the main driver for our sensors and will serve as the brainpowers of the project as it will dictate the actions based on the sensory understandings received from the chair. The microcontroller has input and output ports which will allow us to connect our various sensors and electrical components that our chair will possess. The Microcontroller control units will be tasked with handling of subroutines and interrupt service routines to perform the necessary calculations to determine the proper feedback response.

**Selection**

With our list of needs for this project, we established a list of guidelines necessary to achieve the functionality we are after. One of these guidelines pertains to the amount of general-purpose input and output pins, GPIO, an MCU is able to access. Our design will require many sensors to interface with the microcontroller control unit, so it is dire that we select on option that contains enough GPIO pins. Another guideline is the operational voltage. Most of the embedded sensors currently out in the market are designed for low power environments of less than 5V. We must ensure that the MCU is able to appropriately handle power which will not overload any of the sensors being used. We will talk more about the topic of powering the sensors in the section detailing power consumption. Another important factor is memory. Specifically, RAM and program storage memory. We must ensure that the MCU has enough storage capacity to load our compiled program onto the program storage space. Also, we must ensure the supplied amount of RAM memory is sufficient to run our program without experiencing delays in runtime functionality or loss in data integrity. The last guideline we have for our MCU requirements is the selected CPU. We must ensure that the CPU we select is able to execute the processes in a timely manner. Being able to program the clock and various sleep modes is of high importance as this because it will allow our project to run efficiently while utilizing the low power modes available. This will save power consumption while ensuring that we are able to run our program without any trouble.

**Texas Instruments MSP430**

A microcontroller that is frequently used in Computer Engineering classes at the University of Central Florida, UCF, is the Texas Instruments MSP430 microcontroller. This MCU has ultra-low-power consumption which is convenient for portable applications that run on battery power. This MCU features five low-power modes which ensures we are able to turn off all unnecessary components to save power while not compromising processing speeds. We decided not to go with this option because its board support package, BSP, comprises the Universal Asynchronous Receiver-Transmitter, UART. It is a hardware feature intended to help with the transmission of data while allowing for customization of data format. The reason why we did not like this feature is because it allows us to send a message only after the preceding message has finished which is responsible for latency issues while interfacing with the MCU. Another pitfall from this MCU is the fact that it has a 500 MHz clock speed which is quite fast and would sufficiently keep our program executing flow running favorably, however the faster clock speed could lead to issues with power consumption. For these reasons we decided to focus on other MCU options in order to ensure our power consumption is low while ensuring smooth operation of our program code.

**Texas Instruments F280049PMSR**

Another frequently used MCU in Computer Engineering classes at UCF is the Texas Instruments F280049PMSR. This MCU was used in our Embedded Systems laboratory course and performed well in most of the labs assigned. This system features real time control making it an excellent choice for closed-loop control applications. It has a Trigonometric Math Unit that contains predefined trigonometric operations and functions that are optimized for runtime and the Virterbi/Complex Math Unit. This allows for calculations using complex numbers. The board also features a wide array of self-diagnostic tools such as the missing clock detection circuit and a brownout reset circuit for detecting problems with voltage. This package is managed by an embedded real-time analysis sensor which corrects problems as they occur. A feature that is lacking in this selection is that it only has a single I2C line whiling having two for CAN, SPI, and SCI and 1 for LIN. This is a problem because we have more than 1 sensors that would require communication via the I2C line. This means that we would have significant problems with our project had we chosen to select this MCU.

**TI MSP430FR5739SRHATEP**

The MSP430 offers a breadth of versatility while bringing a familiar development environment and community. Having coursework derived experience with the msp430 it is a strong choice for our platform offering a 16-bit architecture, 24MHz clock frequency, 16 KB of memory, 1kB of RAM, and 16 ADC channels. We also have a number of interface types to choose from as follows: *IrDA, SPI, UART,* & *I2C.* Seeing as this controller typically comes in a tape and reel setup, the package size is quite small, which helps with the minimizing of our PCB area and comes in at roughly 7.72 per unit (under 10 units). Being a part of the TI value line, this controller features low power capability offering and operating range between 2-3.6v. while not an ideal operating range, the sensors being considered would not conflict with this specification.

**Microchip PIC24FJ1024GA606-I/PT**

Looking at other MCU options, we decided to look into the PIC24FJ1024 family. This MCU is known for its hardware real-time clock and calendar system with time stamping that can be used for logging. This feature allows for the reporting of data collected over a period of time. As an example, if we wanted to notify the user of the number of hours they have spent sitting over the course of a month, we would be able to do so with this application. This is a highly beneficial feature as it would allow us to continue to build on our project by implementing new ideas while still allowing us to do the basic functions. Additionally, the sleep and low power modes are modifiable to selectively shut down peripherals to save power automatically when the system sees fit. Another great feature in this MCU is that it supports 3 I2C lines, 3 SPI lines, and 6 UART lines which would provide a total of 12 communication lines for us to connect our sensors to. The chip has a 16-bit architecture which would not provoke any issues with the selected sensors we will be using in our project while insuring that we have a sufficient amount of data bits to store and process our sensory data. A couple of cons for this MCU is that it 5 times and only allows for up to 5 external interrupt sources which restricts the number of features we can implement in our design. Also, the chip lacks native ethernet and WIFI connectivity support which means we must have a separate module to handle the connectivity for our project.

**Microchip Technology SAMA5D2**

The SAMA5D2 chip is another ultra-low-power ARM cortex A5 processor based MPU that we considered for our design project. This MCU runs at processing speeds of up to 500MHz and has ARM NEON SIMD engine which has high processing speeds and is very efficient. This MCU provides many security functions to protect the consumer’s code from being copied by a third-party vendor. While we may have not considered this in the past, this feature is very beneficial if we were to produce our chair in real life as it would prevent a competitor chair maker from taking our technology without our permission. Moreover, the external data transfers are secure preventing hackers from taking the data and using it for malicious means. In our case, it would prevent someone from taking the information gathered from the chair and using it to send malicious code to the MCU to cause it to fail. Some additional features of this MPU are low system cost and high integration with 4-layer PCB with less than 200uA retention mode with fast wakeup. A downside of this MCU is that it included 2 communication lines for I2C which would prove a problem considering we have several sensors that would need to communicate repeatedly with the MCU. Some benefits of the MCU is that it comes with support for LCDs, keyboards, and touchscreens built in. This would be great for add on featured in later additions of our chair such as tracking the amount of time spent typing to ensure the user is spending enough time stretching to prevent arm strain. In addition, to this it came with 7 different types of RAM controllers as well as 10/100Mpbs ethernet support built in. The bus width is 32-bit wide which again presents some issues of underutilization of the data buses again since our sensors are only 16-bit.

**Atmel ATmega2560**

The selection of MCU’s from microchip technology is vast with their acquired *Atmel* portfolio of *ATMEGA*xxx’s. Specifically, the ATMEGA2560U16 offers a 16MHz frequency (internally clocked with an XO), This MCU allows for up to 256 KB of programmable flash which would allow for our project to be of relatively large file size. It contains 2 8-bit timers and 4 16-bit timers for a total of 6 isolated timers. It also offers a total of 86 I/O’s, and 16 ADC channels. These ADC channels are crucial to our design as we will have 10 sensors and may accommodate more should our measurements require a higher degree of accuracy, as defined in our engineering specifications. This MCU employs real-time clock. Another great feature is that it has six low power modes to ensure efficiency and power savings. The MCU allows for four external interrupts which is slightly less than the previously looked at MCU. One of the benefits is that this MCU has a total of 86 general purpose I/O lines which means we will be able to connect several components onto the MCU without any issues. Another plus in this MCU is that it uses the highly favorable 16-bit computer architecture. The chip contains 4 UART, 5 SPI, and 1 I2C serial ports for digital communication. This is by far one of the best options we have as it allows us to simultaneously communicate with the various components we have in our design. Additional requirements that are met with the selection of this microcontroller are the size constraints of our pcb as this is smaller than the TI equivalent. In terms of cost this controller does price higher than the ti at 12.04 per unit (under 25 units). This controller offers an operating voltage of 5 volts with a tolerance of ten percent (plus or minus), this is more ideal for the sensors being considered.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MCU | F280049PMSR | ATmega2560 | PIC24FJ1024GA606-I/PT | MSP430FR2311IPW16R | ATSAMA5D21C-CUR |
| Manufacturer | Texas Instruments | Atmel | Microchip Technology | Texas Instruments | Microchip Technology |
| Program Memory (kB) | 256 | 256 | 1024 | 3.75 | 160 |
| Data Memory (kB) | 100 | X | 32 | 8 | 128 |
| Clock Frequency (MHz) | 100 | X | 32 | 16 | 500 |
| GPIO | 26 | 86 | 53 | 11 | 128 |
| Operating Voltage Range | 1.2 - 1.3V | 1.8V – 5.5V | 2 - 3.6V | 1.8V -3.6V | 1.2V |
| Size L x W (cm) | 33.6 x 33.6 | X | .07 x .07 | .05 x .04 | 1.4 x 1.4 |
| Price (USD) | $10.81 | X | $4.41 | $1.62 | $6.71 |

**Table 5.1:** Decision matrix

**Final Decision**

In the end, our team has decided to choose the Atmel ATmega2560. We decided to use this MCU because it has a vast number of power modes which will ensure that we get the most out of our battery life. It also contains a large number of I/O pins so that we can connect more components to our design without having any issues, such as the need for several multiplexers, or other secondary components required for proper communication between the sensors and user. While we are limited in number of UART connections, we are pleased to have additional SPI and I2C connections which we can utilize to make additional communications with the rest of the components in our design. We also liked that it has a large amount of program memory which means that we would be able to store code that possess a larger number of complex subroutines and multitude of different states to cover a wide variety of situations to ensure that we give the user the most capabilities possible given the data collected. Because the selected MCU has 16-bit architecture there would be no issue writing backward compatible code since all the sensors that will be utilized are specifically for this type of system. The trade of from this design along with the other MCUs such as the Microchip PIC24FJ1024GA606-I/PT is that we would not be able to have a calendar feature which would keep track of usage for a period of time. This would have been a great feature to add as it would allow us to give the user a detailed report of their usage along with suggestions for how they can improve their mobility given the amount of time they spend sitting down. We feel that while this feature would have been great to have, we would like to start our project with an MCU that has a sufficient number of I/O pins to connect all of our components and have sufficient amount of storage space to keep all of our code on the onboard memory. We realized that if we would have gone with the PIC MCU we would have had even more program memory at about 1024 KB which would have allowed us to store all sensory data internally without any problems. This was another reason that made us really consider going with the PIC MCU as the stats for this product seemed to be much better than the ATmega option. One feature that drove our worries away was that the ATmega MCU had significantly more documentation available with regard to programming and pin location. This means that programming all of our components with this MCU would be much more straightforward in comparison to the PIC MCU. We thought this was of high importance because while our team possess great computer engineering skills sufficient enough to code our program, we wanted to find an MCU that was straightforward to program on so that we could focus our time on developing more features rather than spending time troubleshooting errors due to hardware abnormalities

# Piezoresistive Sensor

Based on research and specifications of our project as it relates to sensing the user, the sensor type that meets most of these constraints is the piezoresistive force sensor. To briefly recap, the piezo sensor applies a known voltage across a variable resistance voltage, as pressure (or force) is applied to the material, the resistance will change. This change in resistance is typically linear and can be accurate to within 5 percent. This accuracy typically requires a higher cost. While the sensor can not accurately determine where within its physical housing the pressure is applied, in an array it can very well offer data on weight distribution relative to other sensors. An unintended benefit of this is helping to maintain a degree of user privacy with respect to weight. In the subsequent sections we will discuss some of the sensors being considered as well as include a final decision matrix to determine the best suited sensor for the chair. These considerations will include cost, surface area, load range, drive voltage, and response delay. It is important to note that based on the sensor selected, the driving circuit will be constrained to best suited for said sensor. the driving circuit can be a number of choices including inverting/non inverting op amp or voltage divider. The reason for this is to amplify the change in signal seeing as the resistance change value caused by a load is in the magnitude of tenths of a percent, relative to the initial resistance value.

**FSR-406**

From preliminary research and based on the development platform we begin our considerations with the *Interlink Electronics* force sensitive resistive -406. This is a development kit part designed for actuated load sensing; repetitive task (and range of force). This part offers multiple integration methods which include the following: voltage divider, adjustable buffers, multi-channel digital interface, variable threshold switch, & current to voltage converter. The list provided are methods of application for this device which also amplify the changes in resistance in order to provide usable voltage outputs, which are converted to force estimations. While cost effective this component suffers from the most common issue of piezoresistive sensors, accuracy between sensors can vary as much as forty percent. This is due to several factors which include manufacturing processes, choice of resistive elements, drift (measurement change based on length of time load is static), and signal amplification and processing. The rectangular shape of this sensor would provide optimal coverage when placed in an array which would allow for more precise measurements of weight distribution.

**Spark Fun (SEN-09376)**

the most cost effective choice on the list, and seemingly readily available at the time of writing this brief, the *Spark Fun -09376.* This component comes recommended for development projects based on the Arduino platform. Spark fun also offers a development board for this component which allows for plug and play solutions without the need for an amplifier circuit. The cost of the development board would drive up the price as well as the size of our PCB but would allow for better tuning of the component as well as a controlled test environment.

**TEK Scan (A502)**

In this section we discuss the most costly option, that also provides quantitative superiority with respect to the sensors discussed previously. This sensor offers several benefits including large surface area and low response time. With respect to the component size being a plus, it should be noted that too large of a sensor may result in a lack of precision in our measurements. With this sensor we also have a variable drive voltage option that changes the load range of the sensor; discussed in the datasheet, we see that lower drive voltages offer much higher ranges, this can be tuned according to design requirements which will continue to evolve in the design and test phases. An positive for this component is that the datasheet also provides a recommended driving circuit with a low cost op amp choice. This would reduce development and test time for a driving circuit. This in turn will allow for more time to focus on final design implementation and tuning the sensor range for precise, usable array data. The op amp mentioned is the *Microchip MCP-6004.*

**TEK Scan (A301)**

Another TEK Scan component is the A301. This option offers a circular sensor area, the first for the options being considered. With a diameter of just under point four inches. This sensor offers a maximum load rating of 445N. while smaller compared to the previously discussed sensors, there are a few advantages to its shape. If used in conjunction with a larger sensor, say of a square shape, we can distinguish between predetermined areas or “zones” of a region and more specific subsections within a region. For example, if we define four zones within the base (or seat) of the chair using four large square area sensors, the smaller sensors of the A301 size, could be placed around or in between the zone defining sensors. This would provide data on not only which region carries the most weight (relative to other regions) but also where the weight within the region is distributed. How this data will be used and interpreted will further be discussed in the testing section of this paper.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A502** | **A301** | **SEN-09376** | **FSR-406** |
| Price per unit | 23.24 | 12.18 | 11.25 | 12.99 |
| Surface Area(in2) | 4 | 0.375(diam) | 3.0625 | 2.25 |
| Response time(µs) | ≤ 5 | ≤ 5 | ≤ 5 | ≤ 5 |
| Load range(N) | 44kN max | 444N max | \* | 100N max |
| Drive voltage(V) | Variable | Variable | \* | \* |
| Amplifier circuit (mA) | Multiple | Multiple | Volt. Div. | Multiple |

**Table 5.2:** Force sensor table

**Final Decision**

While cost is an important driver in the decision making process for choosing an optimal sensor, we chose to prioritize accuracy and area covered in our decision making process. This is due to the fundamental nature of these components with respect to the vast number of subsystems that will rely on this device. Based on the varying sizes and availabilities of the discussed items, we have chosen a combination setup of the TEK Scan options, *A502 & A301.* This combination will allow for the establishing of zones within the chair, while maximizing the surface area covered. While the spark fun option would be the most cost effective, the leads are not designed to be soldered, which would be an issue for the permanent connection we hope to make on our PCB. As for the FSR while both cost effective and more solder friendly, we find that the surface area is small, which would require more sensors to provide similar feedback. While our microcontroller selection can support a large number of inputs, these sensors require an ADC to process the input signal before becoming a usable form of data. As such, we wish to not use all the ADC channels on our chip, though the support is there. A benefit of using the two different sensors from *TEK Scan* is that we can average the cost per sensor, which comes to nineteen dollars per device. Since they Are sold in four or eight packs, we will consider ordering one eight pack of each sensor in order to test and develop while also allowing ourselves room for error should receive a damaged component or have a catastrophic test result.

# Development board

In the designing and testing of this device, we find it would be beneficial to have a board upon we can make rapid changes and real time test before moving to a final decisions that end up printed on a board that we cannot modify. As such we will discuss the use of an readily available development board that uses the a closely related variant of the MCU we have chosen.

**ATmega Development board**

In order to simulate proof of concept such as receiving data from our sensor array, processing the data, sending via Bluetooth and controlling the user feedback, we need a board that allows for plug and playability, at a relatively low cost and ease of use. Specifically, we need to supply our *TEKSCAN*  *A series* sensor’s with a 3.3volt supply, receive a signal which will be amplified with an inverting op amp, suggested by the manufacturer. For most of our development and prototype embedded software, we will rely on the *Egeloo ATMega R3* development board. Pictured below this board offers usability of most of the microcontrollers pins. This includes the multiple transmit and receive channels for UART functionality, the 16 ADC channels which we intend to use for our analog force sensors. Through development, which will be discussed in the design section, we found that the MCU’s ADC channels offer variable gain, up to twenty times. this would reduce space on the pcb (as we would not require two op-amp circuits), and the only real cost is a reduction in bit value from 10 to 7. The effects of this will also be discussed in the design section. This board offers a 5V DC input, which requires a wall outlet AC/DC power supply, or a USB-B port which is also used for programming. By using this board we can also reference a functional design while designing our PCB, this can maximize the use of our time so we can focus on other aspects of the project more, should it be required. We can also use this design to make improvements or simply try a different approach as we intend to learn while development a functional product. This board also utilizes a variant of the microcontroller we selected.

# PCB

Designing the PCB will require research and many hours spent in development before we obtain a final product. Below we will discuss the various aspects of designing a pcb including designing, material considerations, and manufacturing. These sections will also include our final decisions on these topics.

# Design software

A major aspect of our project is the design and layout of PCB. In order to design our PCB we first need to have a schematic showing how our components will be connected and situated in relation to one another. This will ensure our ability to maximize the space used on our PCB, while simultaneously minimizing the cost of our printing the board. Below we will discuss the development processes and environments for designing our schematic and subsequent PCB design.

**Eagle**

An *Autodesk* application, eagle offers the ability to design both a schematic and PCB in tandem. Off the bat, this reduces design and development time as we can do clearly see how the digital circuit will look while, weighing the concurrent effects of placing these components on a PCB. Component placement on the PCB affects several factors such as efficient use of space, complexity of traces, and cost of components. These in turn, affect several constraints of our design, if they themselves are not already such. Eagle also offers an industry standard for components, as they are typically vetted, through various libraries and often provided by the manufacturer. This offers better prototyping as the dimensions of components come with a higher degree of certainty. Speaking more on prototyping, being that eagle is more widely used, when it comes to printing our board, and subsequent iterations, we can share the actual eagle file with the board manufacturer. This helps in the event of a small change being needed, the manufacturer may be able to support this rather than going back and forth on the design, or worse yet, receiving a printed board with nonfunctional components. In subsequent sections we will see the schematic and PCB design.

**Easy EDA**

Easy EDA is a web-based application, specifically used for schematic design. This can be used for quick prototyping. While not as widely used as eagle, this application offers the use of user submitted components. While some may be unverified, this can offer a certain degree of clarity in design for components that cannot be easily found through eagle’s libraries. Ultimately for our project, having members familiar with eagle, we chose not to use this application.

**Final Decision**

Ultimately, we chose to use Eagle for development of our schematic and PCB design. It offers, more familiarity as well as more comprehensive library selections with respect to parts.

# Schematic Overview

Based on our preliminary research and hardware selections we have come up with a schematic overview of the system. this will illustrate how we intend to connect our various components including sensors, Bluetooth module, power supply, and several I/O connections. This is illustrated in this section followed by a description of what is presented.



**Figure X: schematic overview**

# Software

In order to meet the requirements of the project, the developers will need to create a mobile application that can pair with the physical chair. A database will be necessary to preserve user data over time. The application will therefore have to be able to communicate with the database and with the PCB on the chair. Outlined in the following sections are the technologies considered for each category and their compatibility. Decisions involving which technologies to use in the development of the application will be derived from the data compiled below.

# Security

<https://www.veracode.com/blog/intro-appsec/importance-application-security-few-benefits-and-risks>

<https://www.upwork.com/hiring/mobile/mobile-application-security/>

<https://www.w3schools.com/sql/sql_injection.asp>

<https://www.veracode.com/security/xss>

<https://newsroom.fb.com/news/2019/03/keeping-passwords-secure/>

<https://www.geeksforgeeks.org/store-password-database/>

The consideration given to data security in application development has increased over the years. The main cause of that heightened awareness being the long line of security breaches that can compromise information ranging from passwords to credit card information to social security numbers. Ensuring that an application is secure is a major concern in software engineering. According to Veracode, having strong security measures in place can improve a company’s reputation, thereby winning over more customers, and can prevent the legal and financial consequences of breaches.

Discussed below is a summary of the measures that Upwork recommends using during development to write a secure application.

* Regarding the portions of the application that will reside on the user’s mobile device:
  + Protect the application code and the API with encryption. If the source code is readable, someone with malicious intent can examine it for weaknesses.
  + Test the source code for vulnerabilities.
  + Design code to be patch-able. Make it easy to fix security vulnerabilities after a breach.
  + Find the balance between the best security and the best performance.
  + Just because an app store says that the app is secure, does not mean that it is 100% invulnerable.
* Security measures for the back end:
  + Consider using containerization to encrypt back end storage.
  + Have penetration tests performed to pinpoint network vulnerabilities.
    - This measure is probably beyond the scope of the project.
  + Encrypt the database and encrypt connections between the front-end and back-end using a VPN, SSL, or TLS.
* Use strong authentication methods like OAuth2, JSON web tokens, or OpenIDConnect.
* “The more data that’s stored locally on a device … the more vulnerable it is.”
* If the data must be stored on the user’s device, encrypt it.
* Avoid storing critical data on the user’s device (ie. credit card number).

**SQL Injection**

A topic that was not discussed in the Upwork article is SQL injection. W3 defines SQL injection as when a malicious user inserts SQL code into a text box that was not intended to receive SQL code. W3 also states that SQL injection is one if the most common ways that web-based applications are hacked.

**SQL Injection Techniques**

W3 discusses what techniques are used to hack an application via SQL injection. The article describes three methods to inject SQL into a text box.

1. 1 = 1

Injecting a constantly true statement like 1 = 1 can manipulate a select statement in such a way that rather than returning a specific row from the database, it could cause the database to return an entire table.

1. “” = “”

Inserting “”=”” into a username or password field could provide the hacker with a list of all of the usernames and passwords since “ “ = “ “ is always true.

1. Batched SQL statements

Entering an SQL command into a textbox that begins with a “;” can allow for the hacker to send batched commands. This is a way to send a second SQL command that is concatenated to the end of the first (and intended) SQL command. An example could be: ; DROP TABLE Students;

**How to prevent SQL Injection**

SQL injection can be prevented by using SQL parameters. SQL parameters are safer than inserting a variable into an SQL command because the SQL engine will check to see if the parameter data matches what the column is expecting. SQL parameters are characterized by the @ symbol.

**XXS Attacks**

XXS attacks (Cross-site scripting) are different from SQL injection in that these attacks do not damage the application, but the user. XXS attacks are possible when the application uses client-side scripting and when the application dynamically uses user data without first validating said data. If a XXS attack is successful, the hacker will gain control of the user’s browser.

In order to prevent XXS attacks, it is important to validate input data and to encode output data.

**Storing Passwords**

Password storage can prove itself to be a major vulnerability in applications lest the developers take steps to protect user’s information in even of a breach. Major websites have made the mistake of saving passwords as plain text. On March 21, 2019 Facebook released a notice that user passwords were stored in plain text which made them readable to Facebook employees. Protecting user data is a high priority in application design. In order to protect users of the project’s application, it was important to the developers to study methods to protect privacy in event of a security breach.

According to a GeeksforGeeks article, storing passwords in plain text is the least secure option. There are several ways to improve the protections for passwords by way of making it harder to recover passwords if a database is accessed illegally.

**Hashing Passwords**

Hashing passwords is the more straightforward method of protecting passwords. The idea behind hashing passwords involves a hash function that maps to the user’s passwords. The hashed version of the password is what gets saved on the database.

But hashed passwords are still vulnerable. Hackers can use a rainbow table to work around hashed passwords. A rainbow table is a table of all of the different hashes of a dictionary of words. The hacker can match the hashed passwords in the database to the rainbow table and derive the passwords from there.

**Salting**

We can improve upon the hashing method described above via salting. Salting is adding an extra string like ‘%67@#\_a’ to the end of a password, then sending the concatenated string through the hash function. This improves the security of the passwords because the hacker would also have to obtain the salt in order to retrieve passwords.

Using dynamic salting is even better than static salting. Dynamic salting involves generating a new salt for each user. The dynamic salt is added with the static salt to the password before it is hashed. Even though the dynamic salt would have to be stored in the database, it is still another step toward more secure password storage. In order to retrieve a password, the hacker would have to get both the static and dynamic salt, then generate new rainbow tables. The salts will increase the time it takes for these rainbow charts to generate considerably.

**Bcrypt and Scrypt**

Even though using dynamic salting will go a long way to slow down a hacker, GeeksforGeeks still recommends using Bcrypt or Scrypt to protect passwords.

# Framework

When choosing a framework for this project, the project members had a set of primary concerns that directed the research. Those primary concerns were:

* Framework must be low-cost, preferably free.
* Framework must have ample documentation and tutorials available.
* Framework system requirements must include both developers’ computer specifications.
* Framework must support needs of the project.

Described below are the best options for the project, given the constraints described above. Due to the impact of a framework on a project, the developers determined that compilation of a list of framework options and the use of careful analysis and elimination of frameworks under consideration would provide the due attention necessary to ensure success of the project.

**IOS-Exclusive Development**

<https://developer.apple.com/documentation/>

<https://learnappmaking.com/ios-development-macbook-pro-good-enough/>

<https://developer.apple.com/xcode/>

To develop for Apple devices, one must use Apple’s IDE, Xcode. Apple is currently on Xcode version 10. The coding languages accepted in Xcode are Swift, C, C++, and Objective-C. Xcode supports three main version control softwares: GitHub, GitLab, and BitBucket. Xcode provides built-in simulations for code, so developer’s will not require an Apple device to test the software’s (isolated) functionality. However, since neither of the two developers have Apple devices, this will strain resources during the integration phase. According to development forums, developing for Apple devices is particularly difficult since there are a lot of regulations on what your application is allowed to do and documentation is hard to come by.

**System Requirements**

* **Windows**
  + **Xcode not available for Windows operating systems. Will require virtual machine.**
* **Mac**
  + **Intel i5 or i7**
  + **4GB of RAM (minimum)**
  + **128 GB of storage (minimum)**
* **Linux**
  + **Xcode not available for Linux operating systems. Will require virtual machine.**

**Android-Exclusive Development**

<https://developer.android.com/docs/>

<https://developer.android.com/studio/>

The development environment for the Android operating system is Android Studio. Extensive documentation available on Android’s website. Android Studio supports the following programing languages:

* Java (all of Java 7, only subset of Java 8) (preferred)
* Kotlin (preferred)
* C/C++ (libraries only)
* Python
* Corona

In the previous iteration of this project, [*Posture Perfect*](http://www.eecs.ucf.edu/seniordesign/sp2016fa2016/g21/), the application was developed for Android in Android Studio.

**System Requirements**

* **Windows**
  + **Windows 7, 8, or 10**
  + **3 GB of RAM (minimum)**
  + **2 GB of disk space**
  + **1280 x 800 screen resolution**
* **Mac**
  + **Mac OS 10.10 to 10.13**
  + **3 GB of RAM (minimum)**
  + **2 GB of disk space**
  + **1280 x 800 screen resolution**
* **Linux**
  + **GNOME or KDE desktop**
  + **64-bit version capable of running 32-bit**
  + **GNU C library 2.19 or later**
  + **3GB of RAM (minimum)**
  + **2GB disk space**
  + **1280 x 800 screen resolution**

**Cross Platform Development Environments**

Instead of writing an application that would be native for either Android or IOS, an app can also be written to be platform independent. Cross-platform frameworks are becoming increasingly popular in the software development sphere. This category of frameworks is known for its cost effectiveness, time saving, and ease of maintenance provided by a single codebase. Described below are the most promising options for cross-platform frameworks to be discussed in detail.

**Flutter**

<https://youtu.be/_ACWeGGBP4E>

Flutter is a new development tool that was released by Google to develop web frontend and mobile application. Being a Software Development Kit (SDK) from Google, this also comes with complete framework, widgets, and tools to allow developers to create applications that can be easy to build and deploy visually pleasing, quick, and smooth mobile apps that can be used for Android and iOS.

One of the many benefits of using Flutter as compared to another SDK, is that it can be cross-platform, being able to run on both Android and iOS, instead of separately. Having to run on one codebase rather than multiple SDK, makes it more convenient and saves time for developers. Having the utilities already built into the system provides very accessible tools such as widgets as customizable design for developers. These beneficial functionalities can be broken down as follows:

* Object-Oriented Language that is based on **Dart**, making it easy to learn
* Built-in Widgets, that can be drawn from its own high-performance rendering engine
* Widgets that are fast, visually pleasing, and customizable
* Ability to create own custom app design, with UI elements available
* Architecture based on React, simple and reactive programming

**Pros:**

The advantages of using Flutter, not only for developers but also users as well, since it speeds up the development process of the mobile application, while also reducing the cost of production. Development teams would greatly benefit from building an app with aesthetically pleasing UI with smooth animation.

* **Code Writing**
  + Mobile Application Development will be faster and dynamic
  + Making changes are more seamless through the process known as **Hot Reload**
  + **Hot Reload** is the process of editing code by updating code into source file while the Dart Virtual Machine stays running
  + Speed of Hot Reload is in the milliseconds to run and test for development and experimenting
  + Designing application made simpler and more comfortable for development process to improve for both designer and tester
  + Less time and building, unlike other native app development which requires rebuilding to retest and experiment the application which can take minutes
* **Multiplatform**
  + Codebase can run on Android and iOS on the same code development
  + Does not need to depend on platform, such as iOS requiring MAC to develop apps
  + Built in library and widgets to create designs for application
* **Testing**
  + Multi-Platform allows developers to save time from running on different system
  + One codebase for an application will work the same on different platform
  + Process Quality Assurance is faster to maintain the level of desired quality of product for close attention to detail on every stage in development
* **Speed**
  + Functions in smooth and fast working environment
  + Seamless transition of mobile application to hanging and cutting while scrolling
  + This section can be further explained in Flutter Technical View
* **Design**
  + Creating widget is made easier with library and customizing existing ones
* **User Interface**
  + Flutter can work with older platform while maintaining functionality of newer supporting systems
* **Minimum Viable Product**
  + MVP is the conceptual process when new product development learns of consumer needs and what requires satisfaction through feedback or collective status

**Cons:**

The disadvantages of Flutter are that since it is new in the developing environment, support is not very abundant for this Software Development Kit. While many libraries are yet to be implemented to Flutter, features are not fully available like any other SDK to develop applications.

* **Libraries and Support**
  + Not every functionality has been featured to Flutter as compared to native development
  + Developers would need to mainly use Google support in order to fully utilize beneficial libraries
  + Even without having full support of libraries like native development, Google has full libraries necessary to develop applications
* **Lack of Continuous Integration Support**
  + Flutter is not widely known compared to other continuous integration platform due to being primarily Google SDK
  + Custom scripts will be necessary to create application using Flutter because of libraries that might not exist at the time for development

Development of mobile application connects many technologies together, with Flutter, the advantages outweighs the disadvantages since it can well support business and development teams rather than having the risk of lacking support. Developing applications with that functions with high-performance while maintaining to be aesthetically pleasing proves to be beneficial while it also runs faster than another SDK to develop such as Android and iOS platform.

Applications have a user end and developer end perspective in the platform that it is used from the native code to communicate between widgets, location, audio, camera, Bluetooth, sensors etc. while keeping in mind the cross-platform nature to run on developing architecture. Mobile applications functions by connecting frontend code to API which communicates with the database, similarly web development runs on similar process. Having cross-platform application is complex since other framework utilizes different mobile and web technologies, with the first cross-platform that was capable to engage were based off JavaScript and WebViews that can now be found on numerous families of framework such as PhoneGap, Apache, Cordova, Ionic and many more.

Viewing applications run with HTML, the structure of the display of the code, which can be then shown on WebView by having it connect with JavaScript to operate and make viewing the application all possible. HTML is what creates the page of the application while JavaScript is how it functions in code. Web development framework like ReactJS have been a favorable library become a main tool for designing web platform.

Flutter provides viewing very much like React Native by taking different methods to perform while avoiding problems when compiling the program from the JavaScript bridge. This bridge acts as the communication between services in native code to connect the realm of JavaScript and Native.

Flutter is Google’s cross-platform mobile development framework. It is free and open sourced. The platform is C/C++ based. The language supported by Flutter is Dart. Flutter’s main selling point is that the developer only has to code up the project once and the program can then run on multiple device operating systems (Android, IOS, Windows). Offers stateful hot reload which allows developers to make modifications to the application while it is running. This speeds up the debugging process. Flutter has a plugin (FlutterBlue) to use bluetooth to communicate with a microcontroller (works for both Android and IOS).

A testimonial from Abbey Road Studios claims that they were able to get their application developed and released in ten weeks with a team of three people.

**System Requirements**

* **Windows**
  + **Windows 7 SP1 or later**
    - **64-bit**
  + **400 MB of storage**
  + **Windows PowerShell 5.0 or later**
  + **Git for Windows 2.x**
    - **Also needs the ‘Use Git from Windows Command Prompt’**
* **Mac**
  + **64-bit macOS**
  + **700 MB of storage**
  + **Access to: bash, curl, git 2.x, mkdir, rm, unzip, which**
* **Linux**
  + **64-bit Linux OS**
  + **600 MB of storage**
  + **Access to: bash, curl, git 2.x, mkdir, rm, unzip, which, xz-utils**
  + **The libGLU.so.1 library**

**PhoneGap**

<http://docs.phonegap.com/>

PhoneGapis a cross-platform mobile development framework developed by Adobe. It is free and open sourced. PhoneGap uses HTML, CSS, and JavaScript for development. PhoneGap maintains access to native technology. It produces hybrid applications via a cloud compiler. PhoneGap supports access to most of a device’s native features (ie. Camera).

**System Requirements**

* **Must have Windows or Mac OS**

While other frameworks were considered, these four appeared to be the most likely to fulfil development requirements for this project. Below is a summary of the data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Android Studio** | **IOS** | **Flutter** | **PhoneGap** | **Appcelerator** | [**Sencha**](https://www.sencha.com/products/touch/) **Touch** |
| **Pros** | IDE is compatible with three main operating systems. | Easier to work with than Android Studio. | Free & open sourced | free & open sourced | extensible & open | Cross-platform (mobile and desktop) |
| Documentation easy to locate. | Big Audience | Fast development with Hot Reload | Cross-platform compatible | Cross-platform compatible |  |
| uses Java | Exclusivity sells | Cross-platform compatible | uses HTML, CSS, and JavaScript | supports 5,000 device and OS APIs |  |
| Used by previous project |  | Supports bluetooth communication (via plugin) |  |  |  |
| **Cons** | IDE is difficult to work with. | IDE is incompatible with the developers’ computers and phones (Mac OS only). | Isn't supported by web browsers | not good for graphics intensive applications | support team is slow to respond | commercial licensing feature is confusing |
| Not cross-platform compatible | The developers do not have experience programming on apple machines. | limited libraries | sub-par documentation | some rewrite needed for *true* cross-platform compatibility | limited themes |
| IDE is difficult to work with | Documentation hard to find. | young | app can render differently on different devices |  |  |

**Table 3.2:** Cross-platform Development Environments Pros & Cons

After evaluating the benefits and drawbacks of writing the application for Apple or Android devices, the developers decided that while it is easier to program in Xcode, Apple does not support the developers’ computer systems. The developers could still work in Xcode but would require virtual machines and an IOS device for testing. Ultimately, it would be more cost and time effective to use a cross-platform framework for the project due to the types of devices owned by the project team and the time constraints.

The developers eliminated Sencha Touch as a possibility due to the restrictions of the free tier. Since the other frameworks provided additional functionality at no cost. Appcelerator was also not selected for additional discussion due to the extra code rewriting needed for the application to work on multiple operating systems. The developers will be on a strict schedule, and re-writing the application to support multiple operating systems is not feasible.

Ultimately, the developers had to decide between Flutter and PhoneGap. Flutter was chosen as the framework for the project due to its speedy development and stateful hot reload which will drastically speed up development. Since Flutter will be used as the framework, the application will be written in Dart.

# **Important Ideas of the Dart Language**

<https://www.dartlang.org/guides/language>

Dart is a programming language that can be easy to learn, compile shared codebase, productive, and ahead of time general purpose tool for developers. The programming language itself is similar to objected oriented language such as C# or Java, also similar to JavaScript (vastly different from Java) that can be used to develop and connect to frontend to backend, meaning user interface and database. Experienced programmers of objected oriented language can Dart as a language much easier to comprehend and learn, and even for less experienced programmers can go forward straight into Dart alone with simpler syntax of code. Aside from the ease of use with Dart, it can also be loose and strong typing, meaning unpredictable producing outcomes of results for loose typing or strict to rules of syntax compilation for strong typing, but both can be applied and simple to move between other languages through Dart.

Sharing codebase across other platforms is difficult, doing so requires portion of the codebase to be ported to other platforms. Dart simplifies the process of sharing codebase due to the nature of being capable compiling natively for both Android and iOS by using the framework for mobile application development tool through Flutter. Since Google supports cross platform, with Dart and Flutter developed by Google, the transition between other platform has been supported all while having the language based on object oriented. Cross platform does not only apply for mobile application, but web and other forms of development. Using Dart programming language through Flutter as the Software Development Kit (SDK), compilation runs ahead of time (AOT) to allow the application working natively and just-in-time (JIT) which develops for local testing. The process of running Dart is much faster as it operates live as code is being changed and no longer having to reload with saving for changes to occur, this gives developers a much faster time to create applications.

Listed below are some of the fundamental ideas that describe how the Dart language functions. Understanding the main ideas that a programming language is built on is important for the developers to successfully transition from a similar coding language (in this case, Java) to the new language.

* If it can be placed in a variable, it is an object.
* An object is an instance of a class.
* Even an integer is an object
* Dart can infer variable type.
* Dart offers support for generic types.
* Like Java, Dart supports top-level functions and variables, static functions and variables, and nested functions.
* Dart does not use the Java keywords public, private, and protected. Instead, if an identifier starts with an underscore, it is private.
* Identifiers must start with either a letter or an underscore.
* If a variable is specified as dynamic, it is not restricted to a specific type.
* Dart offers built-in support for: numbers, strings, ilename, arrays (lists), sets, maps, runes, and symbols.
* Dart numbers are either an integer or a double.
* Strings can be created using either single or double quotes.

# User Data

In order to manage user data, the project will require user sessions. The project will need new users to create accounts and for returning users to sign in. Accounts will allow a user to transfer devices without an interruption in service (ie. Transfer to a new phone). In order to provide user sessions, local storage for the session data will be needed.

In order to save posture data, the developers are considering the use of a combination of an embedded database and a server-side database. The embedded database would reside on the user’s mobile device. The server-side database would reside separate from the user’s mobile device. The server-side database will act as a centralized database to service all users of the application. The embedded database will exist as an instance on the user’s mobile device and will hold only that user’s data. If an embedded database is not used, the application will have to be in frequent contact with the external database which is costly for both the user (in terms of fees on data) and the developers (in terms of fees on access to the central database). The embedded database will synchronize with the central database when convenient (when the user is on wifi).

# Local Storage

<https://www.datastax.com/dev/blog/what-persistence-and-why-does-it-matter>

<https://flutter.dev/docs/cookbook/persistence/reading-writing-files>

<http://www.hwaci.com/sw/sqlite/see.html>

<https://github.com/drydart/flutter_sqlcipher>

<https://github.com/tekartik/sqflite>

<https://www.sqlite.org/whentouse.html>

<https://www.sqlite.org/aff_short.html>

<https://github.com/flutter/plugins/tree/master/packages/shared_preferences>

<https://www.tutorialspoint.com/android/android_shared_preferences.htm>

In order to achieve the objectives of the project’s application, the developers will need to use persistent data. Persistent data is data that has been preserved after the processes that created, modified, or otherwise used this data have terminated. Within the project, the developers anticipate needing persistent data to store settings such as notification preferences, maintain sessions, and incremental data on the user’s posture pending synchronization with a database.

Saving data on mobile devices can be done with either, internal, external, shared preferences, and databases. The options to store app data gives users and developers more resources methods of storage. Protecting privacy is an important factor in storing data, to prevent sensitive information from being widely accessible from other apps. Allowing other apps to access private information requires the consent of the user.

The benefits of having data being internally stored on local device is that the data can be private, making information less widespread across other apps. This makes other apps unable to access said data, to protect privacy of the user, making less direct access. Meaning that the system can provide a directory in the file system to privately store the data and organize the apps necessity. Files can vary from photos, videos, text document, music, downloaded files and such.

While the file stored is kept in the internal storage, uninstalling the app can also remove that save file from the data. This is one of the downsides of storing files in internal storage. This method can be prevented by storing cache directory of the save data into the file system of the device. Applications have private cache directory that functions to prevent lose data in case of the user wishes to remove the app.

Even though cache directory is a viable solution to losing data upon removal of the application, internal storage space can be limit on the device. With limited storage space, Android can delete portions of the cache files to recover space. It can be an option to rely on the system to clean up files, but uncertainties can persist since it is not known to the user on which data can be removed. Maintaining cache by the user is the more secure option so that they can choose which save data to maintain or remove.

To have the Internal Storage function, file objects must use reference to store data by using **FileOutputStream** within the code in order to allow the content to access data only from that specific application. After the data is being stored, accessing this requires internal file directory by using **getFilesDir()** method for the application to retrieve the data from storage. Creating an access point for the directory requires another method, by typing **getDir()** in the code to return reference. To provide file reading preference, the code must include Scanner objects to complete file reading method by using **openFileInput(ilename)**.

**Read/Write Files**

Reading and writing to files to save persistent user data is a very simple method. The path\_provider plugin is a platform-independent method to access common app storage locations on a variety of mobile devices. This plugin provides access to both the cache and the document directory.

Using this method to save temporary user data is easy and there is a comprehensive tutorial available via Flutter to facilitate incorporating the method into a Flutter application.

**SQLCipher**

SQLCipher appears to be very similar to SQLite. It is based on an SQLite database but adds encryption and allows migration from and Android database. Encryption can be implemented on an SQLite database via an extension, the Encryption Extension. Since the developers are also not migrating data from an Android database, SQLCipher seems like it is not the best fit for the project.

**SQLite**

While having a database to store massive data helps in the long run, there is the option of SQLite Database, which are more application specific online storage. Since SQLite is fully available and functional for apps to a lesser degree as compared to a full SQL database, it is required to have knowledge of SQL in order to use SQLite. For mobile application development, using SQLite full features necessary with power and speed as an option for Android and iOS while SQLite also has features of Data Binding.

SQLite is a local database that provides better performance than reading and writing to files. It is free to use and is platform-independent. SQLite use in a Flutter application is supported by the ‘sqflite’ plugin. According to SQLite, it has several advantages over static file read/write:

* Better performance
  + 35% faster than file read/write
  + Load what you need, rather than loading and parsing entire file
* Portability
  + Compatible on all x32 and x64 operating systems (including both little and big endian).
  + Multiple processes can read/write to the same file simultaneously without interference.
* Reliability
  + Continuously-updated to prevent data loss
  + Fewer bugs
* Accessibility
  + Data tends to last longer than the application itself.

SQLite is in the public domain, therefore free to use.

**Shared Preferences**

Another method of storing data is by allowing the user to enter key or value pair of primitive data types, such as Boolean, Float, Int, Long, or String, which are written in XML (eXtensible Markup Language), a language similar to HTML for online storage, with these files that exist throughout multiple sessions. Shared Preferences is an API (Application Programming Interface) that stores information of the user from an interface that access and modify the data of keys and values by hashing, strings of characters that are changed to a lesser and shorter length of data that holds the key and value representing the original stored data. In the case of mobile devices such as Android, the xml file that was transformed can be stored in private directly, while applications can have multiple Shared Preferences of file storage location.

Shared preferences is a method for storing local data in an Android application in <key, value> pairs. The plugin that would permit this kind of storage for a Flutter application is shared\_preferences. This plugin would work for both IOS and Android, but not any other operating systems. Therefore, the developers do not anticipate this method of local storage to be the best solution for the project requirements.

The developers decided to use an SQLite database to manage local storage due to its performance advantages over static file storage and its continuous updates that protect against data loss.

In order to synchronize the embedded database with the central database, a web service will be required.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Read/Write Files** | **SQLCipher** | **SQLite** | **Shared Preferences** |
| **Pros** | Simple and common | Encryption | Cross-platform compatibility | Easy |
| No plugins needed | SQL-based | High-performance and reliability | Lots of resources to use for guidance |
| **Cons** | Slow | Uncommon choice -> limited tutorials | Requires plugin to work with Flutter | Not fully platform-independent |
| Unsecure |  | Requires plugin to add encryption | Requires plugin to work with flutter |

**Table 3.3:** Local Storage Summary

Creating a location for Shared Preferences requires an object that the context of the method can retrieve that object representing the Shared Preferences. This can be done do by typing **SharedPreferences = getPreferences()** object into the code. In the application of Shared Preferences, multiple files can exist and getting the object needs accessing the editor using the **edit()** method to add value and use **put()** method of any data types between Boolean, Float, Int, Long, or String, along with being able to remove the key or value by using the **remove()** method.

# External Storage

Like internal storage, external storage can be used outside of the mobile application or internal storage of the device. This can be done by storing on a computer, or removable SD card and such. Every Android device is capable of sharing data by transferring to external storage and transferring files is also a viable option, such as USB connection to load/copy files from mobile device to a computer. Due to the nature of external storage, it does not guarantee accessibility after the stored data is transferred, since it makes a copy of the file in the system.

Save files in the system maintains on the device while it is active and stored, removal of the application deletes any old files. The files are no longer accessible after the application has been removed from the device. Unlike internal storage, the benefits of having external storage is having a backup of the data, making accessibility of the backup files can be restored once the device re-acquires the application. This is beneficial for the user in the case of having limited storage space in the device, unless external storage can be placed into the device such as SD card to extend space.

Extending storage such as SD card is very favorable to acquire more space, and the installation and removal of the SD card is simple and feasible method of external storage. Relieving the user of concern from storage, since the save data in the SD card can be easily plugged in onto another device.

Creating a filesystem for External Storage is similar to Internal in terms of the development environment, with the difference that External allows removability of the storage device. Providing removability of the device gives External Storage the ability to share stored data to other devices and can be read by any and even all applications.

In order for files to create a directory to allow external devices to store data, **Environment API** must be used to make this possible. The choice of having files be exposed or secure can be given as an option for the user, while developers implement such files on external storage to be saved on private or public space. Using private space allows the user to be able to delete when the app is uninstalled by using the method **getExternalFilesDir()** within the code, while public on the other hand allows accessibility to all files which can be scanned by using the method **getExternalStoragePublicDirectory()** implemented within the code. This gives External Storage more flexibility than compared to Internal Storage, since directory requires parameter to be parsed by the method using **getDirectoryType()** to results a return. Directories are not always set by default, to create location in the storage, the device must be called by implementing the method **mkdirs()** before beginning to save files.

# Database

<https://www.oracle.com/MySQL/>

<https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html>

<https://www.mongodb.com/nosql-explained>

<https://realm.io/products/realm-platform>

<https://github.com/realm/realm-object-server/issues/55>

[https://www.couchbase.com](https://www.couchbase.com/)

<https://github.com/oltrenuovefrontiere/fluttercouch>

Aside from storing saved data on devices and having to transfer files to another device physically by connecting a cable from mobile to a computer, or an SD card to another separate device, a more convenient method to store data would be having a database. Unlike the Internal and External Storage, the use of Database allows accessibility across multiple platform and devices at once. Instead of making copies of save files and transferring to another device, the database can automatically transfer the data whether it can be stored on one device or many, this can make the user experience simpler and less tedious.

The usage of database creates more user friendly and uncomplicated accessibility due to connection of online storage. Android is capable of supporting SQLite databases for storing data, while databases tend to be specific to the app to avoid data to cross over apps without the user’s consent. What makes SQLite beneficial for the developers and users, is the power and speed that comes in full features within a database in correlation to the application.

After deciding on which framework to use for the project, the choices for a database management system were narrowed down to those listed and discussed below. The primary goals when selecting a DBMS were:

* The DBMS must be low cost or free.
* The DBMS must work for both IOS and Android.
  + The purpose of this concern is to prevent the developers from having to create and manage two databases (one for Android, one for IOS).
* The DBMS that will function as the embedded database must be lightweight.

# SQL

A relational database is a database defined by its collection of related tables. Each table has a primary key, and each value within the table is a <key, value> pair. The language used by a relational database is SQL (Structured Query Language).

3.5.4.4.1 Database Normalization

<https://www.tutorialspoint.com/sql/first-normal-form.htm><https://www.tutorialspoint.com/sql/second-normal-form.htm><https://www.tutorialspoint.com/sql/third-normal-form.htm><https://www.studytonight.com/dbms/boyce-codd-normal-form.php><https://www.tutorialspoint.com/dbms/database_normalization.htm>

Database normalization is a technique used to eliminate redundancy in database design. Redundancy can cause a wide range of issues if not removed from the design. Redundancy can cause insertion, update, and deletion anomalies that invalidate data. Redundancy can also increase the size of the database unnecessarily. TutorialsPoint provided an in depth explanation on the 4 main forms of database normalization.

**Requirements for First Normal Form (1NF)**

* Determine which data values are needed. Define the type of data that will be stored for that value (ie. Integer or string). Then group relating values (columns) into a table.
  + Note that each column should only hold one value.
* There should be no repeated groups of data. If a table that should hold transaction information also holds the customer’s address, name, and email the table needs to be split into a customer data table and a transaction records table.
* Each table must have a primary key.

**Requirements for Second Normal Form (2NF)**

* The tables should meet the requirements for 1NF.
* There must be no partial dependencies on the primary keys.
  + Example: Still using the transaction idea, if the transaction data table holds a customerID, customer address, orderID, and order details assume the customerID is a primary key for the customer table and the orderID is the primary key for the order table. The customer address is derived from the customerID. The customer address should be stored in the customer table so it is redundant to have that information in the customer/order table. Since the orderID is the primary key for the order table, it is also redundant and in violation of 2NF rules to have the order details located in the customer/order table.

**Requirements for Third Normal Form (3NF)**

* The tables should meet the requirements for 2NF.
* All transitive dependencies need to be removed.
  + A transitive dependency is when columns b, c, and d contain data that can be derived from the data in column a, but columns a, b, and c are included in a table with other attributes. In order to remedy this, create a second table for b, c, and d with a as the primary key. Then the original table should only contain a and the other data.

**Boyce-Codd Normal Form (BCNF)**

Boyce-Codd Normal Form is a stricter version of 3NF. According to StudyTonight, a table is in BCNF if: For X -> A, X is a superkey.

**User Password Resets**

A standard feature of most applications is permitting users to reset their passwords. In the current culture, most people have a multitude of passwords and will likely need to reset at least once.

**3.5.4.4.2. MySQL**

According to Oracle, MySQL is the world’s most popular open-source database. The project member responsible for the database management has previous experience with MySQL. MySQL can work with Flutter applications via drivers.

According to Datamation, MySQL has the following advantages:

* Data Security
  + MySQL is known for its reliability and data security. It is the DMBS of choice for many popular websites and is frequently chosen for products that perform a high volume of transactions.
* Scalability
  + Datamation claims that MySQL’s scalability is ‘unmatched’.
* High Performance
  + MySQL provides superior performance with its high transactional speed and indexing.
* Up-time
  + MySQL offers ‘24/7’ uptime.
* Transactional Support
  + MySQL provides strong row-level locking to prevent deadlocks.
    - For most projects this should be a primary concern, but there will probably not be a lot of row modification within the application for this project.
* Low Cost
* Open Sourced

**For Establishing MySQL Connection**

In order to establish a connection to a MySQL database via a Flutter application, the plugin, [mysql1](https://pub.dartlang.org/packages/mysql1), will be used. The plugin serves as a driver for MySQL that is compatible with the Dart programming language.

How to Connect to a MySQL database via Flutter:

1. Enter the database connection details into a new ConnectionSettings instance.
2. Use the connect() method to establish a connection. This method returns a connection object that needs to be stored in a variable.
3. Use the query() method to send a query or update to the database. This method will return the SQL results and they need to be stored in a variable.
4. Use a for loop to parse the data in the object query() returned.

**3.5.4.5 NoSQL**

NoSQL is a non-relational database management system. Non-relational databases were designed to cope with modern development trends and to provide higher performance. NoSQL databases are also more scalable than their relational counterparts and have dynamic schemas. Dynamic schemas allow the developer to alter and redefine the schema as the project progresses. According to the MongoDB website, there are several types of NoSQL databases: document databases, graph stores, key-value stores, and wide-column stores.

Document databases involve pairing a ‘key’ with a ‘document’. The document can contain key-value pairs, key-array pairs, or another document.

Graph stores provide data storage in the form of a network. This type of database would work well for networking applications (like Facebook).

Key-value stores rely on the key-value pairs. Values can have variable types. This is the simplest NoSQL database.

Wide-column stores are unique because they store data by columns rather than rows. This type of database is best for handling queries when there is a lot of data.

The NoSQL database that is provided by Firebase is the recommended NoSQL companion to a flutter application.

Of these four NoSQL database types, it is most likely that the project would require a key-value store or a document database. We would not need the graph stores as the application data would not be networked together. The wide-column store would not be a good fit since our application will not be performing massive and frequent queries.

In general, it looks like a NoSQL database is more than we will require for the nature of this project. Since a relational database will probably fulfil all of the development needs, and since one of the developers has prior experience with MySQL, it is unlikely that we will be using this technology.

**3.5.4.5.1 RealmDB**

It is important to note that RealmDB is not free. The lowest-tier of service should be sufficient for this project, but at the price of $30 per month.

According to RealmDB’s website, the following are their selling points:

* Offline First
  + The RealmDB has an embedded and central database model that self-synchronizes when the mobile device has signal.
* Cross-Platform compatibility
* Can be used as a ‘RESTless’ middleware
* Realtime data synchronization
  + Supports reactive applications
* Edge computing
  + Permits developers to cache data anywhere.

Currently, there is no native support for RealmDB with the Flutter SDK. In 2016, a RealmDB developer claimed that it is unlikely to be supported due to the ‘substantial’ nature of the work that would be involved. Due to this roadblock, RealmDB is probably not a good DBMS to use with Flutter.

**3.5.4.5.2 Couchbase Lite**

Couchbase is a NoSQL database management system that provides support for SQL-based queries. The Couchbase website has a page on pricing, but no prices listed. A quote is available upon request. Couchbase Community Edition is free to use.

According to Couchbase, the perks of their mobile database include:

* High availability and disaster recovery
* Security: Authentication and Authorization
* Performance and scaling via adaptive indexes
* Embedded-central synchronization via web sockets
* On-device encryption (Enterprise edition only)

There is a plugin called ‘Fluttercouch’ that offers compatibility between Flutter applications and Couchbase mobile.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SQL** | **NoSQL** | |
| **DBMS** | **MySQL** | **RealmDB** | **Couchbase** |
| **Pros** | Familiar to developers | Offline-first synchronization | High performance and availability |
| Free & open sourced | Realtime data synchronization | Embedded-central synchronization |
| **Cons** | Rigid Schema | $30/month for lowest-tier service | On-device encryption and other features restricted to enterprise edition |
| Must be hosted, which will add to project cost | No support for Flutter SDK | Enterprise edition is costly |

**Table 3.4:** Central Database Summary

Due to the limited return on investment of using a NoSQL database, the developers decided to use an SQL database for the application. Depending on compatibility, the developers hope to use a MySQL database as the central database and SQLite as the embedded database. This is subject to change as the compatibility of the technologies is confirmed.

# Database-Application Interfacing

<https://medium.com/@thedome6/flutter-and-mysql-1d0dc9dfe4af>

<https://medium.com/flutter-community/using-sqlite-in-flutter-187c1a82e8b>

<https://pub.dartlang.org/packages/flutter_sqlcipher>

<https://flutter.dev/docs/cookbook/persistence/reading-writing-files>

<https://github.com/realm/realm-object-server/issues/55>

<https://github.com/hemeda3/couchbase-lite-flutter>

<http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/>

<https://github.com/sqlcipher/android-database-sqlcipher>

<https://developer.android.com/studio/debug/device-file-explorer>

<https://guides.codepath.com/android/Storing-and-Accessing-SharedPreferences>

<https://www.programmableweb.com/api-university/what-are-apis-and-how-do-they-work>

Each combination of technologies have different requirements in order to communicate. Some combinations are already patched together via open-sourced plugins, while others require the developer to write an API (application programming interface). A smaller selection of technologies are designed to work together, requiring little to no effort from the developers. In the table below are the possible combinations of technologies and if and/or how they can be connected. While it is important to select the best technology for the intended purpose, if the database cannot communicate with the application it will cost the development team a lot of time. It is best to avoid a combination of technologies that are not meant to work together, or appear to have a significant workload associated with getting communication established.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | MySQL | SQLite | SQLCipher | Read/  Write Files | Shared Preferences | RealmDB | Couchbase |
| Flutter | PHP | Sqflite package | Flutter\_sqlcipher package | Path\_provider plugin and the dart:io library | Shared\_preferences (provides compatibility for IOS and Android) | Not available - would require custom plugin | Couchbase-lite-flutter (under development) |
| Android | ContentProvider/PHP | Must be custom written | android-database-sqlcipher | DeviceFileExplorer | SharedPReferences interface | Custom API needed | Self-contained within couchbase |

**Table 3.5:** SDK & Storage compatibility

The API/plugin will provide synchronization between the embedded database and the central database. Since the developers intend to use the Flutter SDK and a combination of MySQL and SQLite, the sqflite package will be used to communicate between the application and the embedded database. To connect and synchronize data with the central MySQL database, more research will need to be performed.

**3.5.5.1 Data Binding**

The ability to use SQL allows fetching data from online and delivered to devices, making less effort for the user to retrieve data. Having databases using SQL, uses the technique known as **Data Binding**, which is the process of retrieving date by fragments from the location of the element in the UI and matching the element in the database. Data Binding provides seamless usage of library in the development stage for efficiency and ease of use for mobile application.

* **Support Library**
  + Data binding support its own library that is available to be use for Android 2.1 (API 7) platform to the latest
* **Data Binding Layout**
  + Configuration is required to perform this layout files
  + Files are auto-generated, otherwise without it then it would be needlessly generating random file layout
  + This technique allows layout files to use layout root tag
  + Providing layout root tag allows UI view that can be indicated in the build system
  + Layout root tag allows files to be processed for data binding
  + Without layout root tag cannot be processed for data binding
* **Data Binding Activity**
  + Capabilities of layout file that utilizes data binding loads layout in different way
  + Class that utilizes data binding auto-generates layout file creates default class name that are already set
  + Default names of layout file are capitalizing first letter of each words followed by underscore
  + Code runs faster with data binding because it traverses view hierarchy upon calling for specified view
  + Layout traverse in single runtime because of no more type casting returned views
* **Binding Objects**
  + Binding user object to layout file to assign proper fields directly from layout file
  + Doing so does not require activity to perform when User Object display properties that corresponds to specified fields, such as finding each specific User object
  + Activity to bind objects performs by assigning fields as whole rather than singular field object
* **Data Objects**
  + Can be Java or JavaBeans object that express the evaluated text attribute
  + Objects such as User can bind to automatically generate the Binding class
  + Variable of the layout data such as the user will be generated by creating and binding the object in the Activity (Data Binding Activity)
* **Binding Integers**
  + Having integers can be used in many ways for the user, such as age, weight, height, contact, files, text etc.
  + Data that contain text generally do not accept integers, but can be done by using methods into a string
  + Saving data into string can store the data of a text or integer so that it can be converted to desired output in the layout file
* **Imports** 
  + Data Binding library allows importing classes into the layout file
  + Java allows importing by referencing layout file from a string that is created from a method
* **Data Binding Expressions**
  + Special characters can be complicated to convert into, but storing it into string allows expressions to be stored
  + Java allows expressions to be included which is readily available with useful operators
* **Updating Data Binding Objects**
  + Objects can be displayed by listing and mapping layouts to applications
  + Updating objects does not usually affect the UI, in order to have objects reflect its update into the UI, this requires the use of **Observable Fields**, **Observable Objects**, and **Event Handling**
  + **Observable Fields** – allows accessor methods without needing values directly accessed
  + **Observable Objects** – requires implementing objects to the interface known as **Observable**, or extending it by using **BaseObservable** class
  + **Event Handling** – planning, setting, accessing events by using Method references or Listener bindings to target methods

# **Firebase**

In regards to the [database](https://firebase.google.com/use-cases/), Firebase would help streamline integrating with the front-end. Firebase would make handling sign-up and sign-in functions easier as it manages user sessions and allows the users to sign in and sign up with their Facebook, Twitter, Google, or Github accounts. Firebase would also help with providing custom home screens based on user preferences and user tracking.

**3.5.6.1 Firebase for User Sign In/Up and Custom Screens**

Firebase offers two ways to use it for user accounts and [sessions](https://firebase.google.com/docs/auth/). There is a drop-in authentication feature, FirebaseUIAuth that performs all of the essential functions necessary including password resets. The other option is to use Firebase SDK Authentication. This option allows the developers to decide which features are needed in the application. Those optional features include:

* Email and password authentication
* Authentication via other accounts (Facebook, Twitter, Google, and Github)
* Authentication via SMS
* Custom authentication to work with pre-existing authentication systems.
* Guest/temporary authentication

Firebase also allows for the customization of screens without having to re-publish the application. In order to use this feature, a series of steps must be followed. First, Remote Config must be set up to hold the parameters for the customizable elements. Second, set up Analytics to define audiences for user targeting. Lastly, configure Remote Config to customize elements based on Analytics or user properties.

**3.5.6.2 How does using Firebase impact the database?**

Firebase also functions as a NoSQL [database](https://firebase.google.com/products/realtime-database/). Firebase would also handle offline data management, as well as synching when the user’s device does have signal. Data in a Firebase database is stored as a structured JSON [tree](https://firebase.google.com/docs/database/web/structure-data).

The developers had to decide between using a MySQL database and manually handling sign-in and data management or using a Firebase back-end and having to work with unfamiliar technology. Since Firebase claims to save time for development and would prevent bugs that would be caused if the developers had to handle most of the back-end manually, the developers decided to plan on using Firebase. In order to transition to Firebase, the developers worked to transition the planned MySQL database to a Firebase database by way of [tutorials](https://www.youtube.com/watch?v=ran_Ylug7AE).

**3.5.6.3 Costly Mistakes with Firebase**

An article published by a small Spanish development [group](https://pamartinezandres.com/lessons-learnt-the-hard-way-using-firebase-realtime-database-c609b52b9afb), GreenLionSoft, highlighted some of the issues that can arise when using Firebase. This development group switched to Firebase and over the course of one month accumulated a bill from Google of over $1000. In the article, they described their misunderstandings and mistakes that lead to this huge bill. Summarized below are the recommendations from the development group to avoid such issues.

Tip #1: Use short labels.

Firebase charges based on the amount of data stored ($5/GB) and downloaded ($1/GB). Using short labels for data fields reduced the size of the data stored in the database significantly (reduced storage by 34%) and helped to decrease the storage bill.

Tip #2: Do not enable keepSynch

This was what GreenLionSoft claims was the main source of the hefty bill. The developers *thought* that keepSynch would only synchronize the user’s device with the cloud storage if there had been a change in data. What actually happened, was Firebase synchronized by downloading a fresh copy of the user’s data from the cloud (replacing existing data) every time the application was started. When the use makes modifications, the application updated the data stored on the cloud because uploading data is free. The solution to this problem involved disabling keepSynch and replacing the functionality with a local cache that only existed for the life of the application. This solution did have a downside: If a user manipulates the application in two or more devices simultaneously, the data will not actually synch and update until the application is restarted on both devices.

**3.5.6.4 Users in Firebase Projects**

[Every](https://firebase.google.com/docs/auth/users) application in a Firebase project shares the same user database. User objects represent user account. A user instance is independent from an Auth instance. A user object is made up of a fixed set of properties: a user ID, an email address, a name, and a photo URL. The user object properties can be updated by the user. Any additional user data that needs to be added must be stored separately.

User data is derived automatically from the initial sign in / sign up process. If a federated identity provider is used, the user data will be populated based on that provider’s data. A user instance keeps track of every provider linked to the user.

When a user signs up / signs in, that user becomes the ‘current user’ for that Auth instance. If the page is reloaded, the user is not logged out because the Auth instance persists the user’s state. When a user signs out, that user ceases to be the current user.

The current state of the use can be tracked using listeners. Listeners are notified anytime a relevant event occurs such as: user signs in, user signs out, user changes password, et cetera.

Auth tokens are used by Firebase and Flutter to identify users. Auth tokens come in three varieties: Firebase ID tokens, identity provider tokens, and Firebase custom tokens. Firebase ID tokens are created by Firebase on user sign in. These tokens are signed JSON Web Tokens that uniquely identify the user in a project. These tokens can be used to identify which user is signed in. Identity provider tokens are (usually) OAuth 2.0 tokens created when a federated identity provider is used for sign in or sign up. These tokens are used to verify successful sign in and are then converted into a format that Firebase can use. Firebase custom tokens are created when a user signs in or signs up using the application’s custom authentication service. These tokens are used much like identity provider tokens and exist as JWTs that hold a private key.

**3.5.6.5 How to Implement Firebase Authentication**

There are two methods of implementing user authentication with [Firebase](https://firebase.google.com/docs/auth/): FirebaseUI Auth or Firebase Authentication SDK. As previously discussed, FirebaseUI Auth is a drop-in service that handles all of the authentication process. The Firebase Authentication SDK allows the developers to pick and choose what aspects of Firebase’s authentication system to use. Below is a summary of Firebase’s guide to enabling sign in with both tools.

Implementing FirebaseUI Auth:

1. Select sign in methods and provide any additional data they require. This can be done in the Firebase console.
2. Set up the user interface. The UI can be customized via the FirebaseUI settings.
3. Enable the sign-in flow. This includes importing the FirebaseUI library and picking the permitted sign in methods.

Implementing Firebase Authentication SDK:

1. Select which sign in methods to use and provide configuration data. This can be done in the Firebase console.
2. Implement sign in flows. Federated identity providers may have different sign in flow requirements.
3. Pass the sign in credentials to the Firebase Authentication SDK.

Based on the research gather into backend data management solutions, the developers decided to move forward with Firebase serving as the database as well as many backend functions like the authentication system. If the developers encounter a major issue with Firebase, the secondary plan is to use a MySQL database and will manually handle authentication.

# **Bluetooth**

Connection of software to hardware can be complex and convoluted, in the case of mobile devices, sending and receiving data has abundant methods and among many ways to connection, Bluetooth is a manageable option. Mobile devices support Bluetooth to allow wireless connection of exchange of data between the software and hardware thanks to the provides API and framework between mobile and Bluetooth.

**Android Bluetooth**

Bluetooth can support multiple platform; the Android platform can support the network stack to allow the communication between devices. The wireless exchange of data uses the framework to access and connect from Bluetooth with the Android API, to enable point-to-point features. Android has a built-in function to allow multiple Bluetooth devices connectivity all at once by using the API in the Android system application for scanning, query local adapter, establishing channels, and transferring data while all with the capabilities.

* Scan Bluetooth Device
* Query local Bluetooth Device
* Establish RFCOMM channels
* Connectivity to other device service discovery
* Transfer data of another device
* Manage multiple connection

Bluetooth pairing requires data to be transmitted between devices in order to enable the communication process. Security levels can be set to place precaution of connectivity between devices by pairing and bonding, doing so requires permission to request, accept, connect, and transfer data. Setting up applications to take Bluetooth connectivity requires permission of the location of devices to be scanned and can also be used to gather information for the process to transfer data. To set permission between connectivity application and device, declarations must be made for an access point to initiate the device to be enable for discovery by declaring **ACCESS\_COARSE\_LOCATION** or **ACCESS\_FINE\_LOCATION**.

Application allows profile for any device connected via Bluetooth, giving it an identity to distinguish from other devices wirelessly since Android 3.0 with the full support of the API. Profiles comes with interfaces such as Hands-Free, that supports mobile device to connect to headsets, vehicles, audio, and health device. Proxy is setup to connect objects of the profile data to establish transmission.

* **Headset**
  + Bluetooth API gives profiles for headsets in applications, which can be used in mobile devices or tablets
  + Profiles are given a proxy that is provided in Android to control devices
* **Audio**
  + Profiles can also be provided for audio devices that contains high quality audio
  + A2DP (Advanced Audio Distribution Profile) is used for devices that can switch connection between audio devices
* **Health**
  + Profiles for health devices such as heart-rate monitors, blood meters, thermometers, and scales etc.
  + Since Bluetooth can exchange data between connectivity, profiles can provide saved data

Health Device Profile (HDP) was introduced in Android 4.0 (API level 14) to provide applications and devices for constant communication in monitoring health. The API to use HDP can includes key classes to be described in interfaces. HDP has main concepts for health departments that provides beneficial services to patients.

**Bluetooth Pairing**

It is important to know which device can be compatible for communication to ensure that connectivity can be supported. Device that can be supported will send discovery point for scanning in local area for any enabled Bluetooth devices to request transmission and information exchange. The process of pairing Bluetooth with enabled devices is often called discovering, inquiring, or scanning. Device that are within range of discovery will respond by requesting and accepting information, to exchange name, class and unique address. After pairing is made between connection for the first time, request can be set automatically as the device will be remembered for the information already paired which is then saved. There are distinguishable differences between being paired and being connected.

* **Paired**
  + Being aware of existing devices that share an already established link for authentication and encrypted connection
* **Connected**
  + Sharing an RFCOMM channel that are capable of transmitting data between Bluetooth devices
  + Pairing will be established once initiation occurs to encrypt the connection with API

To pair devices in order to make it discoverable, the devices must be querying to return objects to bond and connect, that way it can be recognized, and data will bring up the profile for specific device. The querying process will then acquire the name and MAC (Media Access Control) address to initiate connection begins as an asynchronous process to return Boolean value in the scanning inquiry.

**Bluetooth Low Energy:**

Running Bluetooth on devices can drain battery life faster than normal while being active, by keeping connectivity and communication between other devices. To save power for the user in order to make usage of mobile device for longer periods of time thanks to latest Android API connection. The new android 4.3 latest update for API (level 18) platform, provides support built into the device for Bluetooth Low Energy (BLE) for connectivity of the API.

**Bluetooth Communication to Android:**

Bluetooth modules and Mobile devices such as Android can communicate with the use of microcontrollers to integrate between software and hardware. Using microcontrollers to implement the communication between software and hardware are main factors in electronics for creative prototypes of experiments like the Bluetooth to mobile device. Bluetooth is a useful tool for short-range wireless communication to other electronic devices in developing hardware connectivity along with mobile application for software implementation using microcontroller. While there are multiple microcontrollers exist for electronic experiment, the prime device used will be the ATmega 2560 for the practice of the experimentation.

Implementing Bluetooth connectivity with mobile device is uncomplicated since writing the code does not use libraries but rather just simple and only needs the serial transmission and setup methods. Locating the serial transmission is simply done by connecting the right wire to the pins of the microcontroller assigning to that specific command in the code along with the baud rate for transmission. The serial transmission should be set at the proper baud rate of 9600 as the default, any higher baud rate can cause the transmission to be more susceptible to noise that would disrupt the connectivity. Once the Bluetooth connects to the microcontroller, the next process would be sending and receiving data to devices that have paired with the Bluetooth.

Establishing the communication with the Bluetooth to the mobile device requires both to be enabled for discoverability in order to have both be paired. On mobile devices such as Android, enabling the option to scan other devices should display multiple devices and detect the Bluetooth module. Pairing request is send to the module from Android, and a pop-up option window will display on screen with password input. This form of connection is based on models for client-server that waits for connection request or sends out the request to the server, meaning the Bluetooth device.

**For Establishing Bluetooth Connection with Flutter**

In order to establish a Bluetooth connection via a Flutter application, the plugin, [Flutterblue](https://pub.dartlang.org/packages/flutter_blue) will be used. Flutterblue is a plugin that works for both Android and iOS devices. Flutterblue provides support for scanning for bluetooth devices, connecting to devices, and interacting with those devices.

How to Connect to a Bluetooth device via Flutter:

1. Create an instance of flutterblue
2. Use the scan() method and a listener to obtain the scan result. This will look for bluetooth enabled devices nearby. A scan can be terminated with the cancel() method.
3. Use the connect() method and a listener to connect and verify connection to a device. The connection can be terminated with the cancel() method.
4. Once the connection is established, use the readDescriptor() and writeDescriptor() methods to communicate with the paired device.

# Password Recovery

A [satisfactory](https://auth0.com/learn/password-reset/) password reset model makes it easy for a user to reset their password (less than 1 minute) and allows a user to reset their password while keeping their data safe.

If the Firebase authentication system is used, the developers will not have to handle password resets as Firebase will manage those functions. If a different authentication system is used, the developers may have to handle password resets.

According to [Troyhunt](https://www.troyhunt.com/everything-you-ever-wanted-to-know/).com, the two best and most secure way to facilitate password resets is to:

1. Have the server generate a temporary password and email that password to the user.
2. Email a unique URL that handles the password reset.

The article also highlights the importance of never using the method of ‘reminding’ the user of their password. In order to use this method, the passwords must either be stored in plain text or they are encrypted. This is not the best way to store passwords.

* + 1. Push Notifications

Push notifications will be used in the application to remind the user to stand and walk around every ninety minutes. There will be different approaches to performing push notifications depending on if a MySQL database or a Firebase database is used.

In order to perform push notifications with Firebase, the application will need to use Firebase Cloud [Messaging](https://medium.com/flutterpub/enabling-firebase-cloud-messaging-push-notifications-with-flutter-39b08f2ed723) (FCM). This service has a free tier up to a threshold that this project will not reach. There are two types of push notifications via FCM: data messages and notification messages. For this project, we will be using notification messages. Implementing this strategy will require slightly different code for Android and IOS devices.

It does not look like there is support for performing notifications with a MySQL database.

# **Standards & Constraint**

Needs intro

<https://www.w3.org/TR/mobile-accessibility-mapping/>

[https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo#lazy-programming](https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo)

<https://www.w3.org/standards/webdesign/accessibility>

<https://www.w3.org/WAI/fundamentals/accessibility-principles/>

<https://uvision.co/technology/how-to-get-your-website-or-app-ready-for-gdpr-and-other-privacy-standards/>

# **Distribution - App Store Requirements (to be a subsection in Standards)**

The [easiest](https://support.google.com/googleplay/android-developer/answer/113469?hl=en) and safest way for a user to obtain the application is to download and install the application via an application store like Google Play Store or the Apple App Store. In order to have the application available on these devices, the application must be submitted to the app stores for approval. There are a series of privacy and safety requirements that applications hosted on these stores must meet. These requirements will serve as project constraints.

Note: According to ClearBridgeMobile.com, the [Apple](https://clearbridgemobile.com/how-to-submit-an-app-to-the-google-play-store/) App Store has much stricter requirements than the Google Play Store.

**4.1.1 Google Play Store**

In order to [upload](https://clearbridgemobile.com/how-to-submit-an-app-to-the-app-store-updated/) an application to the Google Play Store, the developer must have a developer account. Obtaining a developer account involves a one-time $25 fee.

How to upload an application to Google Play, according to ClearBridgeMobile.com:

* Make sure application functions correctly. Fix any bugs.
* Ensure application does not exceed maximum file size. If it does, files can be modulated to an extent.
  + Android 2.3 and up: 100MB
  + Android 2.2 and lower: 50MB
* Enter application details, graphics, and screenshots.
* Upload files
* Perform alpha and beta tests
* Publish

Note: After ‘publishing’ via the developer account, there will be a small delay before the application is visible to others (a few hours).

**4.1.2 Apple App Store**

The [application](https://developer.apple.com/app-store/review/guidelines/) must pass Apple’s App Store Guidelines. Below is a summary of guidelines from Apple that are relevant to the project’s functionality.

Safety Guidelines

* Apps should not contain offensive or inflammatory content (ie. discrimination, animal abuse, pornographic material, prank phone calls).
* Apps that contain user generated content must have a filtering and review plan.
* If application will be in the kids category, must not contain targeted ads or in-app purchases
* Must not promote/allow physical harm (ie. inaccurate medical information, remind users to check with a doctor)
* Provide a method of bug reporting.
* Protect user data.

Performance Guidelines

* The application should be complete (including metadata and placeholder text).
* Do not use the app store for beta testing.
* Make sure metadata is correct and up to date.
* Do not include undocumented features in the application.
* Ensure device compatibility.
* Use power efficiently.
* Declare additional equipment/peripherals needed to use the app.
* App must only use public APIs.
* Apps should be self-contained in their bundles.
* Application must not transmit a virus.
* App must be compatible with IPv6 addressing.
* Apps must not alter the functionality of the device.
* Do not submit apps with empty or sample banner ads.
* Must request user permission to record user activity.

Business Guidelines

* Applications should not require the user to perform additional tasks such as rating or reviewing the application.

Design Guidelines

* Application design must not be ‘copycat’.
* Application must fulfil a useful purpose and not be ‘creepy’.
* Application must not repurpose keyboard shortcuts.

Legal Guidelines

* Application must protect user data.
* Application must comply with privacy best practices, ‘applicable laws’, and the Apple Developer Program License Agreement
* App must include link to privacy policy in metadata.
* App must get user consent to collect data.
* Application should only access data necessary to perform core functionality of the application.
* Application cannot ‘use, transmit, or share’ personal data without user consent.
* Data cannot be used for multiple purposes without notification and consent.
* Application must not store health information in iCloud.
* Application must not include ‘borrowed’ content.
* Application must abide by the Developer Code of Conduct

Given that the application meets the requirements summarized above, the application should be permitted to distribute itself via the Apple App Store. According to ClearBridgeMobile.com, these are the steps that should be taken to submit an app for review to the Apple App Store.

How to submit application to Apple Store:

1. Create iTunes Connect app record
2. Build application.
3. Archive application.
4. Application must run and pass iTunes Connect validation tests.
5. Upload app to iTunes Connect
6. Submit app for review
7. Release application.

# **Chair electricity Standards**

Electricity is very dangerous in the project that we have been in because it can kill people in a second. Safety in smart is our main purpose because we do not want to put people live in danger. The PCB will be under the chair and rely on some sensor that will be in the back of the chair, those sensors will have wires connected to the board that has electricity flowing on it. As we know the smart chair will be cleaned by using water, disinfectant and many more stuffs that can put life in danger, we want to use the maximum safety necessary to protect the chair specially when they use it outdoor and it is raining. As we know water and electricity do not mix, we will have the safetiest wires and material in the

# **Power Supply Standards**

According to POWER SUPPLY SAFETY STANDARDS, AGENCIES, AND MARKS which are responsible for the principle of electrical safety standards, the national and international safety ask to use the one of the standards that requires in your country and certify by the government to prevent fire, electric shock and injury should meet their requirements. Below are the standards that describes our power supply requirements for the safe use of the electric equipment, batteries.

|  |  |  |
| --- | --- | --- |
| Standards | Description | Details |
| IEC 60950-1 | Safety of Information Technology Equipment. | Intended to prevent injury and damage such as electric shock, fire, dangerous temperature. |
| IEC 60065 | Safety of video, Audio, and similar electronic apparatus. | Intended to protect against fire, electric shock and injury, electronic equipment and communication. |
| IEC 62368-1 | Audio, Video, Information and communication technology equipment | Standards that currently govern companies marketing audio-visual products, computing and communications equipment in North America. |
| IEC 60601-1 | Safety of Medical electrical equipment. | Covers the basic safety and essential performance applicable to medical electric equipment including surgical, monitoring, hospital devices and been revised so many times. |
| IEC-61010-1 | Safety of measurement, Control and Laboratory equipment. | Requirements for measurement, control and laboratory equipment. It protects the electrical shock, fire and burns injury. |
| UL 1310 | Safety, Standards requirements for 2 class power units. | Covers indoors, outdoors that use 2 power supplies and batteries. Uses for residential and industrial. |
| UL 600799 | Standards that use for explosive atmospheres. | Covers electrical equipment that use for explosive atmosphere like construction, gas, combustible dust. |

**Table 4.1:** Power Supply Major Safety Standards

* 1. **Flutter Development Standard**

For the front-end development, the developers will abide by the Flutter code standards to promote efficient code and ease of reading.

**4.5 Disability Standards for App Development**

For the project, the development team will abide by accessibility standards set by W3. These standards are in place in order to improve access to technology for users with disabilities. Described below are measures that can be taken to increase the inclusivity of an application.

* Provide alternative text for images.
* Provide a keyboard alternative to mouse/finger movements. If a user cannot use a mouse, they may be using a speech-to-text device that can mimic keyboard strokes.
* Provide transcripts for audio.

**4.6 User Privacy Standards**

User privacy is becoming increasingly important to application developers, application users, and governments alike. Governments are beginning to enact privacy standards to protect their citizens. The European Union enacted privacy standards that impact every application that has a single EU citizen’s data stored in it. Abiding by these privacy standards is important for both the user’s privacy and the company’s (or application’s ) reputation and global availability. It will also reduce penalties for noncompliance.

**4.7 Mobile Application Standards**

Mobile application has become widespread in global market for software and technology for consumer usage and innovative design, and as part of core modeling to develop application now available on small devices, standards are set to create a universal understanding on how mobile devices function. Great potential can be achieved through mobile devices that allows users access to web, app, planner, media, and information that can support daily life, while developers can expand the capabilities and enhancing cross device design. Standards can be set in principles, interfaces, patterns, and guidelines to developing mobile applications.

**4.8 Firebase Standards**

To ensure efficient use of resources, Firebase has a set of recommended practices. They have been summarized [below](https://firebase.google.com/docs/projects/learn-more#best-practices).

All variants of the same application should be placed on the same Firebase project. However, if the variants use different build bundles, then they should be separated. There is no limit on the number of applications that can be placed within the same project, but there is a limit on the number of OAuth 2.0 client IDs.

If two or more applications do not share the same data, they should be located in separate projects to prevent the complications and privacy concerns that accompany multi-tenancy.

**4.8.1 Principles**

The overall Principles can be numerous ways to create a platform that can document and function natively with individual operating system on each device. Mobile platform can greatly benefit consumer to allow a system that create an experience to unify others with an ecosystem to support. Other platforms can also support a unifying experience with its own ecosystem, with the day to day usage of devices enhancing the user to fully utilize such feature, the mobility of a small platform that goes straight into pockets makes convenience the key leverage to the capabilities. While mobile devices can be identified as a phone, applications can exist in cross platform to scale functionality among ecosystem created with similar experience. The overall Principles can be broken down as follow:

* **Platform**
  + An important factor is documenting the necessity such as the pattern and components that comes in the native operating systems in the application such as Android and iOS
  + Designing platform that is native to the device should remain consistent with the operating system and guidelines to focus on optimal quality
  + Native operating system to the platform can improve and evolve with new guidelines to enhance interface
* **Benefit**
  + Importance of consumer needs is a prioritize focus in developing mobile application that offers capabilities to support day to day needs
  + Designs should be able to bring together multiple users to unify an experience within an ecosystem across any device
* **Device**
  + Device should have capabilities such as touch, voice, pressure, location tracking, accelerometer, notification, etc.
  + The design of the application should revolve around being able to be utilized on the device that can benefit not only the screen but beyond that
* **Scalability**
  + Mobile device can go beyond than just a phone that can talk and text, modern phones can search through the web, applications to plan and schedule, gaming platform, media for music and video, and many more
  + Capabilities on the phone can also be scaled on tablets with the same features and if not more, and grow beyond simple screen usage
  + Although challenges can be met by scaling interface between web and tablet designs of the patterns and guidelines
  + There can be similarity between mobile devices and full web applications on a computer that proves to be difficult to be used similarly

**4.8.2 Interface Platform (Standard)**

Another challenge met by developers is the compact and small size of **phones** which are typically less than 7 inches, that can only display more limited amount of information and screen usage as compared to the web on a computer. To compromise with the smaller size display of mobile devices, the fundamental design must include primarily the necessary information for users to search and view so that the device does not overload. Since mobile devices are the most convenient platform to access information for users, it can also be a useful device for business to quickly access and continue work that the user is not usually and readily available at an office.

Other than phones, another mobile device would be a **tablet**, which are greater than 7 inches of mobile display for the user interface that would allow more space for design and usability. While alignments do not have to be the same from tablets to phones, having larger interface display would greatly benefit by having more information for the fundamental design to include in the platform. While tablets have the designs that tend to have an experience similar to desktop to search the web, plan and schedule, workspace, and media, which is also available on phones, the main functionality of tablets can be considered as a hybrid device.

**4.8.3 Patterns and Guidelines (Standard)**

Mobile applications have design patterns that follows to fit on small platforms to navigate through content while feeling quick and easy to use, simplicity should be the key factor in creating applications. Navigating content should have transition phase to display through applications to feel smooth and organize without feeling cluttered and disarrange.

**4.8.4 Industry Implementation (Standard)**

Establishing set of rules and guidelines for platforms to follow becomes a universal practice for industries to develop mobile applications while also evolving on new and inventive ideas for technology of the software. There are multiple set of standards the industry implements.

* **Visibility and Timing**
  + Notification is an important feature that allow users to view information which companies provide in applications to be relevant for the framework
  + Accessing data also collects and transmit important application features while prioritizing user accessibility with privacy
  + Sensitive information such as financial data will be held accountable on the company to provide and ensure secure utilization
* **Security and Data Retention**
  + Importance of security in transmitting data must be for legitimate purpose and should be access in the application only unless it is required to do so
  + Sensitive information should be given as option that lets the user to allow their data to be collected to avoid risk
  + Developers must implement this option in order to avoid danger of data being leaked to give users security in experience
  + Rules and regulations must be met for developers to publish the application that follows the app store and platform terms and services
  + Sensitive data must be stored on a time frame where it has the chance of being deleted after user no longer needs data stored in servers
  + Respecting the user privacy keeps their interest in the service of the application with procedures holding accountable of private information
  + De-Identification is another solution to ensure security of data by deleting previous data to reestablish identification by hashing and linked back to original source of user or device
* **Enabling Security Measures**
  + Applications are susceptible to security risks of accessing or transferring data from individuals to another that requires careful attention
  + Testing and solution must be taken for security measures for implementing retention policies for safeguarding data
* **Data Encryption**
  + Encrypting data allows authentications of the user personal data through transmission to provide protection
  + Proper utilization of server by avoiding SSL/TLS and other forms of communications of transmitting data
  + Sensitive information such as email, address, username, and password, must be encrypted to maintain authentication
* **De-Identification**
  + Like encryption, changing identification on multiple efforts makes it difficult for data to be at risk since it does not link to particular result
  + Numerous identification elements require scrambling encrypted data to have comparable link to that element
  + This method ensures higher security measures to terms of privacy to avoid possible risk while maintaining retention
* **User Authentication**
  + Logging in and out requires authentication of the user by inputting either email, address, username, and password for the session with validation
  + Implementing mobile client to ensure validation of the user to access their data from the server database where it is stored with sensitive encryption protocol
* **Accountability**
  + Responsibility of the company gives user the respect to hold the application development team accountable for managing the safety and privacy of their data
  + Protections of sensitive data should be fully integrated to the application to maintain the policy of company duties of terms and services
* **User Feedback**
  + Maintaining good relations with user can be provided by having feedback for opportunities to improve flaws of bugs of the application
  + Companies gain positive note from questions, contacts, queries, or complains of application interface and usage
  + Fixes and patched can follow from feedback to highlight functionality to bring more efficient solution

# **Initial Design**

# Software

The developers will begin initial design of the application. This phase of project planning will help the developers identify any additional research or technologies that will be needed to meet the project goals. In order to meet the project requirements the developers deemed it prudent to use user stories in order to develop a more detailed list of functions the application must possess in order to better direct development.

*As the application is developed, the designs dictated in this document are subject to change.*

**User Stories**

* The user should be able to install the application on their device.
* The user should be able to create an account.
* The user should be able to sign in.
* The user should be able to pair the application with a chair.
* The user should be able to see data from the chair displayed on the application.
* The user should be able to control how the application sends notifications.
* The application should be able to connect to the PCB.
* The application should be able to send commands to the PCB.
* The application should be able to insert data into the database.
* The application should be able to read data from the database.

**Software Requirements Derived From User Stories**

* The application will meet system requirements of at least one type of mobile device.
* The application will allow account creation.
* The database will allow for an insert operation into the users table.
* The application will allow for user sessions via sign in.
* The application will connect to a PCB via bluetooth.
* The application will read posture data from the database and display the data.
* The application will save user preferences regarding notifications.
* The application will send commands to the PCB.
* The application will insert data into the database’s posture table.
* The application will read data from the database’s posture table.

# Application and Database

# User Interface



**Figure 5.1:** User Account Creation and Login

Note: PK indicate that this piece of information will serve as the primary key in the database.

Note: Diagram made on draw.io

# General Design



**Figure 5.2:** Refined User Interface

# Widgets

# Database Design with MySQL

The central database will need to store user data and posture data. Below is the initial design for the database.

**Users Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | uid | email | firstName | lastName | password | dynSalt |
| Data Type | int | string | string | string | string | String |

The uid will be the primary key for this table.

**Combo table**

|  |  |  |
| --- | --- | --- |
| Field | uid | pid |
| Data Type | int | int |

The uid is a foreign key. The pid is a foreign key. The primary key will be <uid, pid>.

**Posture table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | pid | datetime | cogX | cogY |
| Data Type | int | datetime | double | double |

The pid will be the primary key for this table.

The purpose of the database in this project is to save the user’s posture data. The database will also prevent total loss of data if a user needs to move their account to a new device (phone). MySQL will be the database management system for this project.

The database needs to hold two sets of information: user’s account data and user’s posture data. This will require two tables as demonstrated in the figure below.

**Figure X: ER Diagram generated by MySQL Workbench**

**Request permission to use this diagram!!!**

The posture data is being saved in two fields, cogX and cogY. These fields will represent the center of gravity of the user through the perspective of the axes front-to-back (cogY) and left-to-right (cogX). The purpose of the created\_at field is to allow the data to be ordered temporally in order to display graphically.

The user data includes the user’s login information (email and password). The securityAnswer field is in place to facilitate future resetting of passwords.

# **Database Design with Firebase**

According to an [article](https://howtofirebase.com/firebase-data-modeling-939585ade7f4) from How To Firebase, the database functions best when the data is as normalized (shallow) as possible. Nesting data within the database will increase the cost and decrease the speeds of downloads. One important comment from the article highlighted the importance of finding a balance between normalized data and reducing extra downloads. The article used a transaction log as an example: If email addresses are frequently being pulled when transactions are pulled, it might be better to include a copy of the email address in the transaction log. This results in only the transaction log being pulled rather than both the users and the transactions. This method also increases the scalability of the data.

When retrieving data, it is better to use a child\_added event than a value event. Value events return all nodes in an unsorted JSON object and will return all nodes every time a single value is changed. This is less efficient than a child\_added event which fires for every existing child and fires again when a child is added. Also, since child\_added fires for each child, it can work with orderBy modifiers, which means that the developers will not have to manually sort the data.

The article also recommends using queues to handle server processes. This highly scalable method involves adding any processes like changing usernames or transactions to a queue so that they are all handled in order and processes do not get lost.

The image below is a visual to represent the structuring of the NoSQL Firebase database. The securityAnswer field was omitted because Firebase will be handling password reset/recovery. The users data is restricted to what Firebase automatically gathers.



**Figure X. A Visualization of the NoSQL Firebase database.**

# Embedded Software

# Hardware

# Bluetooth Module

There are multiple kinds of Bluetooth modules that can be used for electronic experiments, but the main focus is primary between Bluetooth 2.0 ERD Modules and the Bluetooth 4 / BLE modules to compare which would contain more features, functionality, simplicity, and implementation. The number of pins can vary between the different version from 2.0 and 4.0, with the most common pins are EN, WAKEUP, STATE, BRK, KEY, or LED that can be available across any of these versions.

# Bluetooth 2.0 EDR Modules

From this type of Bluetooth module, the most commonly used modules are the HC-05 and HC-06, with both being capable as slave device.

# HC-05

This Bluetooth module uses SMD module that is based on BC417 models and has a design to be used for transparent wireless serial connection that goes along with the setup of Bluetooth SPP (Serial Port Protocol). This module can cover range up to 9 meters, or 30 feet of signal that can function as master or slave. This module uses full duplex data transmission, meaning it has capabilities to transmit both direction of sending and receiving end of signal carrier at the same time. The pin configuration for the HC-05 includes Enable / Key, Vcc, Ground, TX – Transmitter, RX – Receiver, State, LED, Button. When running the experiment to send signal of the module, it will have default setting to be used when coding the microcontroller, with the follow:

**Default Setting**

* **Name:** “HC-05”
* **Password:** 1234 / 0000
* **Communication:** Slave
* **Mode:** Data Mode
* **Baud Rate:** 9600, 8, N, 1
* **Command Mode Baud Rate:** 38400, 8, N, 1
* **Firmware:** LINVOR

**Specifications**

* **Voltage:** 4V – 6V
* **Current:** 30mA
* **Range:** <100m
* **Compatibility:** Serial Communication (USART) and TTL
* **Standard:** IEEE standardized protocol
* **Signal:** Frequency-Hopping Spread Spectrum (FHSS)
* **Operation:** Master, Slave, Master/Slave mode (simultaneously)
* **Usability:** PC and mobile with Bluetooth capability
* **Baud Rate:** 9600, 19200, 38400, 57600, 115200, 230400, 460800

Using the HC-05 for experiments can add full-duplex wireless functionality towards projects that can communicate across other microcontrollers, from mobile devices and PC to interface with applications. The USART and the baud rate at 9600 makes the process simpler to implement and configure the module through command mode. This module is widely used in electronic projects for wireless data transfer to and from PC and mobile but does not have functionality to transfer multimedia content. The benefits of using this module is having two operating modes, where one Data mode is capable of sending and receiving from other devices with Bluetooth functionality, the other data mode is AT Command mode to allows changes made onto the default setting using the pins. This module is simple thanks to the operations of Serial Port Protocol (SPP), while only needing to use up to 5V to power up from the pin with ground, this allows it to enter to Command mode, unless the left was left idle for a while then the setting goes back to default which is Data mode.

# HC-6

This is very much like the HC-05 Bluetooth module with the only difference being that the HC-06 is only a slave device. This makes this module more suitable for simpler projects for wireless data transmission that only requires the Bluetooth end for slave mode, with the same range as the HC-05 at 9 meters, or 30 feet. This module can be used for short range data communication that is setup from two different microcontroller or even any system with built-in Bluetooth. The transmission speed of this module can go up to 2.1 Mb/s while also being the most affordable Bluetooth module to use for projects and very flexible to implement. The signal of transmission uses frequency hopping spread spectrum (FHSS) like the HC-05, this type of signal avoids interferences across other devices in order to achieve full duplex transmission that was explained in HC-05 for directions of sending and receiving signal carrier. The pin configuration is identical to the HC-05, with the exception of Enable, LED, and button. The configurations include the Key, Vcc, Ground, TXD, RXD, and State.

**Features**

* **Bluetooth version** 2.0 protocol standard
* **Power level:** Class2(+6dBm)
* **Band:** 2.40 GHz, ISM Band
* **Receiver Sensitivity:** -85dBm
* **USB version** 1.1/2.0
* **Modulation Mode:** Gauss Frequency Shift Keying
* **Authentication and Encryption**
* **Voltage Operation:** +3.3V – +6V
* **Temperature Range:** -20ºC to +55ºC
* **Current:** 40mA

**Advantages**

* Optimal choice for short range wireless communication
* Wireless distance of less than 100 meters
* Simple to implement interface and communication
* Very affordable cost for module that functions all types of capabilities in present
* Very low consumption of power to maintain and capable of being powered by mobile battery to function
* Has capabilities to be interfaced with nearly all microcontrollers and microprocessors so long as there is a UART

Since the interface communication is done through the UART have the module to be functional, sending data to mobile or receiving data from the module can be establish through any device. Powering the module can be connected using the +5V pin from the standard regulated power supply with the UART establishing an interface, while the connection to exchanging data from RXD to TXD and vice versa.

# Bluetooth 4 / BLE Modules

BLE (Bluetooth Low Energy) modules can be beneficial for saving power, and like the HC series, this also uses SMD module based on the Bluetooth SOC (System On Chip). The type of Bluetooth comes in 2 versions from the HM-10 series, the S version and C version that does not need pads from the bottom connector (USB connection). Both the S and C versions have 26 pads instead of the usual 34 from other modules, making this the cheater option to produce and manufacture. BLE itself is not considered an upgrade from the Bluetooth Classic modules, but instead it uses a different system and intention of usage.

# HM-10C

# HM-10S

# BLE Link Bee

This module has capabilities to support master and slave switch with a long transmission range of up to 60 meters throughout any free space, which is one of the many benefits of Bluetooth 4 modules for latest design. The design for this module is more compact than the HC and HM series, that has voltage regulator integrated into the MCU that can support both 3.3V and 5V.

# Implemented Bluetooth Module

Microcontroller to Bluetooth connection works as a master and slave device, where the microcontroller is the master, and Bluetooth is the slave for the device. This means that the master device gives commands to the slave device to communication with all functionality such as exchanging data. The Bluetooth has a set standard range for the wireless connectivity, it uses **short-wavelength UHF radio waves** that has a band range of **2.4 to 2.485 GHz**. The technology built-in Bluetooth can be embedded in any device, while the module communications with mobile devices from building personal area network (PAN), the range it can be reached from the band would be approximately 9 meters, or 30 feet. Each pin on the Bluetooth module serves a purpose, with the Enable/Key, Vcc, Ground, TX, RX, State as pins to connect with features for functionality.

* **Enable/Key**
  + The enable pin serves as a switch for the Data Mode and AT Command Mode
  + The switch for Data Mode is setting low power for the module
  + The switch for AT Command Mode is setting high power for the module
  + Normally at default, the module is set to Data Mode for low power
* **Vcc**
  + This is the pin that powers-up the module and serves as the supply voltage
  + Connection for supply voltage can go up to 5V
* **Ground**
  + The ground pin of the module to regulate the voltage to zero potential
* **TX**
  + This pin is the Transmit Serial Data on the Bluetooth module so that it connects to the RX on the other device to acts as the receiver
  + Serial data that gets transmitted is being received through the other end to give out data which goes to the pin to the receiver UART of the microcontroller
* **RX**
  + This pin is the Receiver Serial Data on the Bluetooth module so that it connects to the TX on the other device to acts as the transmitter
  + This acts to request serial data that is given to this pin, while the other end of the connection transmit from the microcontroller
* **State**
  + Serves as the connection to the LED on the Bluetooth module from the board itself
  + Functions to check if Bluetooth is functional and flash or blinks for feedback ensuring proper working condition

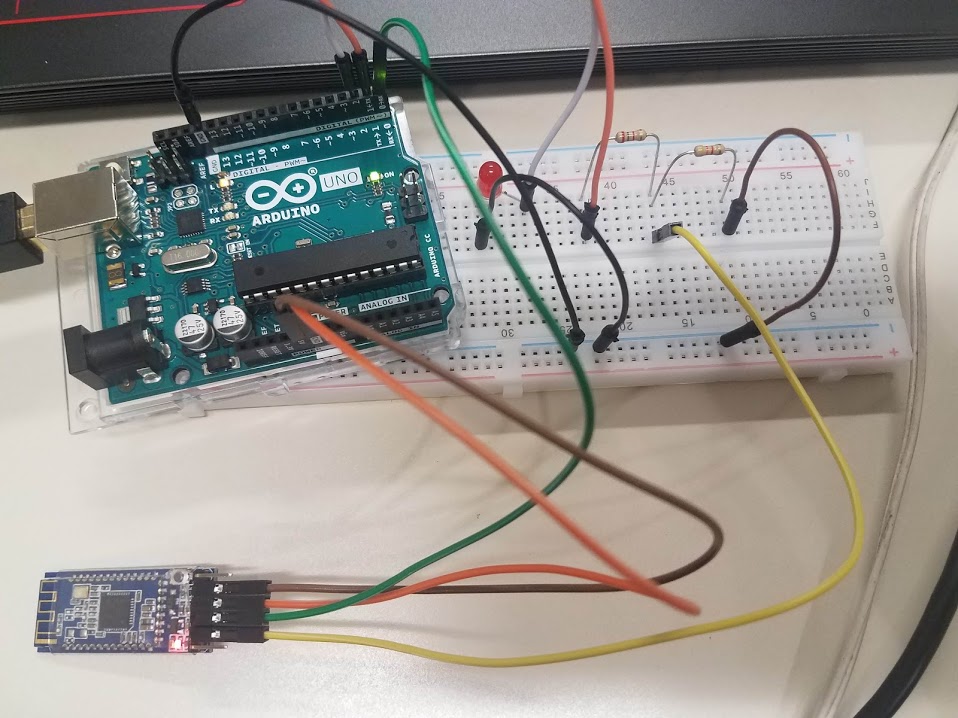
Sending sensor data from Bluetooth to communicate is an uncomplicated process that requires few simple tools to conduct the experiment, which are microcontroller, Bluetooth module, USB cable, breadboard, and sensors. The microcontroller used for the experiment will be the Arduino Uno, the Bluetooth module used will be HC-05, and all this will be connected to any dedicated computer to code the project. The experiment will require multiple procedures to follow right down from the wiring, Bluetooth connectivity, the coding process, transmitting data, and troubleshooting to have the electronic devices function, and the steps are as follow:



* **Circuit Setup**
  + The Bluetooth module (HC-05) will be connected by having the voltage lines through a voltage divider, that way the module does not get burned and fry the circuitry
  + The wire will be connected to a line, but because of the voltage divider, 5V may not be sustained in the line to pass through so it does not damage the module
  + On safer line to connect, the module can be connected to a 3.3V line, unless more power is required for other inputs
* **Bluetooth Connection**
  + The next step is pairing the Bluetooth module with any computer with built-in Bluetooth capabilities, otherwise an external dongle can be implemented
  + Connecting the Bluetooth can be done on the computer by going to the settings, locating the device, detecting the module, and pair with a pass code
  + Going to the main settings of the computer can be followed as such: Control Panel, Hardware and Sound, Add a Device
  + Detecting the HC-05 and typing a code of 1234 or 0000 to begin implementing to finish the pairing
* **Arduino Code**
  + From the sketch provided of the design to implement the experiment for proper functionality
  + Finding the correct ports of the board to connect the Bluetooth module can be found in the settings of the IDE
  + Once the correct port and board is selected, the TX and RX pins can be disconnected after uploading the code
  + Otherwise an error will occur when the COM port is busy if not disconnected after the code is uploaded
  + When uploading the code is successfully completed, reconnect the TX and the RX pins to the original port
* **Receiving Data**
  + After the code is uploaded to the board, in this step, the power source can be used to connect the board
  + After the code is uploaded and power source is used, USB cable can be disconnected from the board to the PC
  + This exhibits the step that no longer needs the PC to power the device and confirms that uploading the code is finished to use the power source on the board
  + Also showing that the sensor can function by remotely gathering the data and ready to transfer to the PC through Bluetooth
  + Notification setting can be set in order to see if Bluetooth and sensors are functional
  + Locating the COM port of the Bluetooth module can be set and display on screen on whether the data of the sensor is being send
  + An incoming and outgoing COM port can be set to signify sending and requesting data
* **Troubleshoot**
  + Data can be fully functional when the serial can be monitored on screen to show notification is set
  + In the case of data having delay and takes more time to successfully connect, this means that an error may occur in the process
  + Bluetooth communication problems occur due to signal interference, and can be avoided to ensure secure and firm signal
  + Rechecking the TX and RX pin connection to ensure functional communication
  + Changing to different voltage so that power can be properly distributed, connecting Vcc to 3.3 V and not 5 V
  + Loose connection can be an occurrence and stops the Bluetooth module from working, this can be seen on the module by checking if the LED blinks continuously to indicate if it is being powered
  + Disconnecting and reconnecting of the Bluetooth module to check by following the procedures previously listed and see if communication can function properly
  + Bluetooth module can be out of range from the PC
  + These steps followed can ensure simple Bluetooth experimentation

# Bluetooth Breadboard Experiment

To test out the experiment to determine whether the Bluetooth module is fully functional, the module is then connected to the microcontroller to be programmed. The pins used on the Bluetooth module will be the Vcc, GND, TX, and RX, wired to the microcontroller to begin testing to ensure that it gives off signal to be detected to other devices.



# PCB



# Proximity Sensor



**Figure 6.1: Project Management Visualization**

# Milestones

**Hardware**

* + 1. Circuit Design & software simulation including calculations and filter/gain stages
    2. breadboard simulation with test parts
    3. Design PCB & simulate
    4. Order final parts
    5. Order PCBs
    6. Solder the parts on the PCB and reflow
    7. Test PCB

**Software**

1. Storyboard the user interface
2. Design Database
3. Learn how to create a mobile application for android and learn how to use local storage.
4. Create Database and populate with sample data.
5. Create skeleton UI and get interaction with backend.
6. Algorithm to analyze the data from hardware and to recommend posture adjustments.
7. Communication with hardware to send commands to LEDs and receive sensor data.
8. Finished version of UI.
9. Fully functional application backend.

**Integration**

1. Send and receive data via Bluetooth
2. User setting controls LED’s
3. Read sensor data from PCB’s

# **Appendix**

# Works Cited

Yoga and Stretches

<https://www.verywellhealth.com/yoga-for-back-pain-supine-spinal-twist-297346>

<https://www.doyouyoga.com/5-yoga-poses-to-ease-lower-back-pain-30237/>

<https://www.yogajournal.com/poses/sphinx-pose>

<https://www.yogaoutlet.com/guides/how-to-do-thread-the-needle-pose-in-yoga>

<https://www.yogaoutlet.com/guides/how-to-do-thread-the-needle-pose-in-yoga>

Agile and Scrum

<https://www.agilealliance.org/agile101/>

Center of Gravity Relating to Posture

<https://www.verywellhealth.com/human-center-of-gravity-296568>

Database Normalization

<https://www.tutorialspoint.com/sql/first-normal-form.htm><https://www.tutorialspoint.com/sql/second-normal-form.htm><https://www.tutorialspoint.com/sql/third-normal-form.htm><https://www.studytonight.com/dbms/boyce-codd-normal-form.php><https://www.tutorialspoint.com/dbms/database_normalization.htm>

IOS Development

<https://developer.apple.com/documentation/>

<https://learnappmaking.com/ios-development-macbook-pro-good-enough/>

<https://developer.apple.com/xcode/>

Android Development

<https://developer.android.com/docs/>

<https://developer.android.com/studio/>

Flutter Development

<https://youtu.be/_ACWeGGBP4E>

Dart Language

<https://www.dartlang.org/guides/language>

PhoneGap Development

<http://docs.phonegap.com/>

Sencha Touch Development

<https://www.sencha.com/products/touch/>

Data Storage

<https://www.datastax.com/dev/blog/what-persistence-and-why-does-it-matter>

<https://flutter.dev/docs/cookbook/persistence/reading-writing-files>

<http://www.hwaci.com/sw/sqlite/see.html>

<https://github.com/drydart/flutter_sqlcipher>

<https://github.com/tekartik/sqflite>

<https://www.sqlite.org/whentouse.html>

<https://www.sqlite.org/aff_short.html>

<https://github.com/flutter/plugins/tree/master/packages/shared_preferences>

<https://www.tutorialspoint.com/android/android_shared_preferences.htm>

<https://www.oracle.com/MySQL/>

<https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html>

<https://www.mongodb.com/nosql-explained>

<https://realm.io/products/realm-platform>

<https://github.com/realm/realm-object-server/issues/55>

[https://www.couchbase.com](https://www.couchbase.com/)

<https://github.com/oltrenuovefrontiere/fluttercouch>

Connecting Database to Application

<https://medium.com/@thedome6/flutter-and-mysql-1d0dc9dfe4af>

<https://medium.com/flutter-community/using-sqlite-in-flutter-187c1a82e8b>

<https://pub.dartlang.org/packages/flutter_sqlcipher>

<https://flutter.dev/docs/cookbook/persistence/reading-writing-files>

<https://github.com/realm/realm-object-server/issues/55>

<https://github.com/hemeda3/couchbase-lite-flutter>

<http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/>

<https://github.com/sqlcipher/android-database-sqlcipher>

<https://developer.android.com/studio/debug/device-file-explorer>

<https://guides.codepath.com/android/Storing-and-Accessing-SharedPreferences>

<https://www.programmableweb.com/api-university/what-are-apis-and-how-do-they-work>

Application Security

<https://www.veracode.com/blog/intro-appsec/importance-application-security-few-benefits-and-risks>

<https://www.upwork.com/hiring/mobile/mobile-application-security/>

<https://www.w3schools.com/sql/sql_injection.asp>

<https://www.veracode.com/security/xss>

<https://newsroom.fb.com/news/2019/03/keeping-passwords-secure/>

<https://www.geeksforgeeks.org/store-password-database/>

Software Standards

<https://www.w3.org/TR/mobile-accessibility-mapping/>

[https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo#lazy-programming](https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo)

<https://www.w3.org/standards/webdesign/accessibility>

<https://www.w3.org/WAI/fundamentals/accessibility-principles/>

<https://uvision.co/technology/how-to-get-your-website-or-app-ready-for-gdpr-and-other-privacy-standards/>

Previous Version of this Project (PERMISSION REQUESTED)

<http://www.eecs.ucf.edu/seniordesign/sp2016fa2016/g21/>

Storage:

<https://developer.android.com/guide/topics/data/data-storage>

<https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/>

Data Binding:

<https://www.androidauthority.com/data-binding-in-android-709747/>

Bluetooth:

<https://developer.android.com/guide/topics/connectivity/bluetooth>

<https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>

<https://examples.javacodegeeks.com/android/android-bluetooth-connection-example/>

<https://www.tutorialspoint.com/android/android_bluetooth.htm>

Flutter:

<https://www.thedroidsonroids.com/blog/flutter-in-mobile-app-development-pros-and-cons-for-app-owners>

<https://flutter.dev/docs/development/tools/hot-reload>

<https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514>

<https://flutter.dev/docs/development/ui/widgets>

<https://medium.com/asos-techblog/flutter-vs-react-native-for-ios-android-app-development-c41b4e038db9>

Minimum Viable Product:

<https://www.agilealliance.org/glossary/mvp/#q=~(infinite~false~filters~(tags~(~'mvp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)>

<http://ask.leanstack.com/lean-startup-fundamentals/what-is-a-minimum-viable-product-mvp>

Reactive:

<https://medium.freecodecamp.org/an-introduction-to-functional-reactive-programming-in-redux-b0c14d097836>

<https://reactjs.org/>

Dart:

<https://www.dartlang.org/>

<https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf>

<https://medium.com/@mswehli/why-dart-is-the-language-to-learn-of-2018-e5fa12adb6c1>

Mobile App Standards:

<https://medium.com/blueprint-by-intuit/native-mobile-app-design-overall-principles-and-common-patterns-26edee8ced10>

<https://devtechnosys.com/know-about-industry-standard-for-app-development>

Bluetooth to Mobile Device:

<https://softwarehut.com/blog/how-to-communicate-small-arduino-device-with-android-phone-via-bluetooth/>

<http://www.electronics-lab.com/project/arduino-communication-android-app-via-bluetooth/>

<https://deepbluembedded.com/bluetooth-module-hc05-interfacing-pic-microcontroller-tutorial/>

# 7.2 Permissions Requested / Acquired

Posture Perfect

Status: Permission Request sent via email.



Draw.io diagrams permission granted: <https://desk.draw.io/support/solutions/articles/16000042494-what-are-the-usage-terms-for-diagrams-produced-by-draw-io->