Convert an Industry Leading Native Mobile App to React Native

Project Plan

Team Number: sdmay19-02

Client: Buildertrend

Adviser: Mai Zheng

Victor Amupitan –– Chief Engineer of Design

Lucas Kern –– Executive Meeting Facilitator

Michielu Menning –– Lead Report Manager

Kyle Nordstrom –– Co-Team Lead/Meeting Scribe

Francis San Filippo –– Scrum Master

Walter Seymour –– Co-Team Lead/ Team Communications Leader

Team Email: sdmay19-02@iastate.edu

Team Website: https://sdmay18-01.sd.ece.iastate.edu

Revised: September 27th, 2018 / Version 2

Table of Contents

0.1 List of Figures

0.2 List of Tables

0.3 List of Symbols

*We will fill these later as they are needed

0.4 List of Definitions

- **BT:** Buildertrend, our Client.
- **Components:** Unit of code that deals with a specific feature or functionality. Components break apart the project into smaller subtasks.
- **Component Based Development:** A software development process that emphasizes the separation of concerns throughout the project.
- **DRY programming:** Don't repeat yourself programming (ie duplicating logic)
- **Front-end:** The part of development that deals with converting data into a graphical interface for users to interact with.
- **Jake:** Director of Software Development at Buildertrend.
- **Native:** Software that is developed for use on a particular platform or device.
- **React:** A javascript library for building user interfaces.
- **React Native:** A framework for building native applications with React.
- **React Router:** Specifies the components that will be displayed with certain routes.
- **Redux:** A predictable state container for JavaScript applications.
- **SaaS:** Software as a service
- **Software Architecture:** High level structures of a software system.

1 Introductory Material

1.1 Acknowledgement

We would like to give a special thanks to the following members of Iowa State and Buildertrend for the assistance they have provided throughout our project.

- Rich Kalasky
- Daric Teske
- Mai Zheng

1.2 Problem Statement

The problem that Buildertrend currently faces is that the mobile application that they use could be created and maintained more efficiently. They currently are updating and maintaining two applications, one for iOS and one for Android. This makes the software harder to maintain. Maintaining two applications that are in more obscure languages is costing Buildertrend time and money. They are looking for a way to save money, time, and make the overall development process easier in the future.

Buildertrend had the idea to recreate the application with React Native. A React Native application can solve their problems by maintaining one app instead of two. With React Native, the JS code can be translated into Native code for both iOS and Android. There only needs to be one application, and this means BT can drastically cut the number of resources needed to maintain and update the application. Furthermore, by transitioning into React components, the project code base will be much cleaner. Reusing and managing components makes for a more efficient and maintainable development process.

1.3 Operating Environment

This is a mobile application, and the main users are construction workers. This means the environment that the application will be used in may vary greatly. Because these construction sites are sometimes in remote locations that may have a poor internet connection, the efficiency of the application is key. It is also possible that the users may not always have direct access to the device itself. This is in part due to the nature of the industry. With massive amounts of manual labor being done on-site, it may not be safe or viable for them to be on their phone actively checking the application. Likewise, construction sites are not the only place the application will be used. The application is used by many project managers off-site or in an office setting. This means that it will be used in places with more predictable conditions. Therefore, the application needs to perform consistently and serve as a productive tool for the construction industry.

Overall, there does not seem to be any direct safety hazards, other than the general dangers of using a mobile device; e.g. texting and driving. When using this application, the environment may impact the quality of the product. For this reason, we must consider the variety of uses and environments the application will be subjected to.

1.4 Intended Users and Intended Uses

There is a multitude of different types of users that will be working with the application. These include:

- Construction Managers - These are the people who will be working with projects, but also have capabilities that the average user may not have.
- Construction Workers - Common users that will have less access than the managers.
- Clients - People who paid for the construction job. They are able to track their jobs and request changes.
- Sales Employees - They will be able to track the projects that they are assigned to. They will also have access to internal features.
- Developers - Buildertrend Developers will also have access to the application. They will have their own specific features to oversee the product.

The construction managers use of the application will be to manage the project and to manage the people under them. These uses will be to keep the construction projects on schedule and to make sure employees are up to speed on project expectations. The workers themselves will be able to use the app to handle tasks they have completed, clock into work, and be in contact with the manager. In addition, the construction managers will be able to use the application to track leads and handle financial issues.

The client's main use of the application will be to oversee the progress of their project(s). They are also able to contact the construction manager regarding any inquiries about change orders. They are capable of communicating fluidly with the project manager in case any legal agreements or documents require a signature.

Sales employees will be able to use the application to track their project managers and make sure that they are not having any issues with the application. Above the sales team, the developers will have access to virtually any functionality. This is to account for any testing and adjustments they may want to explore.

1.5 Assumptions and Limitations

- Assumptions
    - Users will have access to a mobile device.
    - The primary user will be construction managers and workers.
    - The backend of the application, which will not be provided to us, will handle all security measures.
    - The backend will handle any financial transactions or legal transactions between the builder and clients.
    - The backend will give us data in an efficient and clean way.
    - The backend of the application will handle all the accepted file extensions.
    - No one has access to the backend code besides us and employees of Buildertrend.
    - All margins and styles will be standard to cell phones.
    - The Code will be tested by the QA at Buildertrend.

- Limitations
    - The speed of the product relies on the phone it is on and some old phones may not handle the software as well.
    - Usage of the application depends on a solid connection to the internet.
    - Many of the developers in our group do not have much experience with React, Redux, and some other tools.
    - Clients pay a lot of money to use the product, so the product must be at a high standard.
    - The product must be completed by the end of the second semester.
    - The functionality of the application must match the functionality of the current application.

○

1.6 Expected End Product and Other Deliverables

Throughout the development cycle, there will be a multitude of minor and major deliverables. Some of these delivered items will include weekly reports, schedules, and documentation. Moreover, our main deliverables are as follow:

- **Initial Prototype** - This will be a fully functioning React Native project. It will have a functioning menu, and the user will be able to pull up the list of components. There may be a few subcomponents completed, but the main purpose is to establish user movement throughout the application. This will lay out the foundation for the following deliverables and will serve as a framework for further development.
- **Structured Prototype** - This deliverable will be a project management application with the core features implemented. It will not be a completed version of the initial prototype, but rather a partially finished version. It will have all the same functionality and structure, but some of the components will still require completion. In order to fill the necessary features, the components within the existing application will be ordered by priority.
- **Final Prototype** - This is the final product that we will present to Buildertrend. Upgrading from the structured prototype, it will have all the features the original application has. We will have all the existing components built out and fully functioning. There is potential that we may add new features as the client and we see fit.
- **Documentation** - This will be the API for the public functions that we use while making the product. This will be used for the people who will be working on the project in the future. It is also be used for developers at the company who may have questions about the application.

We estimate completion and delivery of this project to be May 2019, the end of the Spring 2019 semester.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The desired outcome of this project is to have a smooth application running on both Android and iOS that hits on all of the functional requirements, nonfunctional requirements, and deliverables.

The internal aspects of this task are to completely recreate two applications using React Native, a newer technology. By successfully merging two different projects into one, we are helping Buildertrend eliminate an unnecessary project. By having both Android and iOS devices using the same application, it ensures that all the features are uniformly applicable across the entire user base.

After this is complete, every construction manager that utilizes this industry-leading software will be able to accurately and successfully construct their guzzling projects.

2.2 Functional Requirements

Our application will have different levels of clearance depending on the role of the user. As typical each user will have an account to log into. This will be handled on the backend by Buildertrend. They will then have access to all or some of the following features:

1. **Daily Logs** - Record of logs with messages to stakeholders which can include information such as dates, attachments, tags, and weather conditions.
2. **To-Do's** - Tasks the user can make for stakeholders. Can include text, attachments, assignees, and reminders.
3. **Schedules** - The schedule includes calendars, Gantt charts, list views, agendas, workday exceptions, and functionality to add a task to the schedule.
4. **Change Orders** - Change order can be created, updated, deleted and updated. Includes any information that would be on a real change order.
5. **Selections** - Component where job site managers can make selections about a job site. Can view them by list, grid, hierarchy, category, location, and allowance. Users can CRUD selections.
6. **Warranty** - Warranty section where users can CRUD warranties on associated jobs.
7. **Job Info** - Job info section shows the details of the selected job and allows users to CRUD jobs.
8. **Quick Actions** - Quick actions is a part of the navbar that links to frequently used components. These include - Adding photos, daily logs, to-dos, messages, and documents.
9. **Photos** - Component that allows users to upload photos and store them in custom albums.

10. **Documents** - Component that allows users to upload documents and store them in custom folders. Users can search by a document name.
11. **Videos** - Component that allows users to upload videos and store them in custom folders. Users can search by videos.
12. **Messaging** - Messaging allows users to send messages and comments to other stakeholders of the project. Also, users send surveys and notifications.
13. **Financial** - Where users manage all financial aspects of job sites. This component includes functionality for CRUD ops on budgets, Bill/POs, owner payments, and bids.
14. **Directory** - Users can store contacts for stakeholders and anyone they want a contact form. List view of all contacts with a search bar for filtering.
15. **Sales** - Sales component includes sections for users to record lead opportunities and lead proposals. A calendar view of sales activities can be viewed.
16. **Misc** - Miscellaneous features include a contact us page, logout button and links to the full site.

2.3 Constraints Considerations

The non-functional requirements of our product include:

- **Responsive:** The mobile application should be user-friendly by providing quicker results and reducing lag.
- **Reliability:** The mobile application should be reliable regardless of network connections because some users might not have access to the internet.
- **Energy efficient:** The application should be energy efficient because users might be in remote locations with limited power supply.
- **Security:** The application contains user data so security measures should be taken. Security measures include authentication, authorization, and encryption.
- **Performance:** The application is expected to have close to real-time results so it should be efficient with handling user requests.
- **Availability:** The application should be available at any point as long as the phone is active.
- **Maintainability:** The application is to be built in a modular way using a component architecture to make it easily maintainable
- **Data Integrity:** The application is to be built taking data integrity by displaying and sending accurate data and maintaining consistency.
- **Usability:** The application should be user-friendly and easy to use without a detailed explanation of how the application works.

The constraints that are we considering in this project are as follows:

- **Human Resources:** Our team is a fixed sized team with each member having their own strengths. In addition to classes, all of us are working jobs on the side. The balance between school and work will be essential throughout the semester.

- **Technical:** Our client has already chosen the technology they want the product to be developed with - which is React Native.
- **Time:** The product must be completed before the final presentation of the Senior Design class.

Standards for software engineering relate to data security and safe transaction handling. Our application will not handle any sensitive data such as credit cards which would use the PCI compliance standard. However, many standards can be used as a framework to improve the overall quality of our application. ISO/IEC/IEEE 29119 is a standard for software testing which outlines how to test software for any organization. ISO/IEC 12119 is a standard for how packages are delivered to the client. ISO/IEC 9126 details how the quality of a software product is determined. We should keep this standard in mind at the end of the project so all stakeholders have a guideline for evaluating product quality.

While developing this project, we will be following the IEEE Ethics and Compliance which includes the IEEE Code of Conduct and IEEE Code of Ethics. These standards, in summary, describe ethical rules that should be followed in the development process and with the product developed. In order to upload IEEE's code of ethics, we must ensure the product we deliver is of the utmost quality and sees to benefit all stakeholders involved. Also, within the team, ethics can be applied:

- While developing this project, team members have to respectful of each other.
- The product should not be harmful to the users

2.4 Previous Work And Literature

2.4.1 Existing Works

Our project is focused on the development of an application using React Native. There are some applications that provide similar functionality, but the focus of the project is to rebuild the application using a new framework (Reference #9). The work we are providing is for employees of Buildertrend to continue the development of their application. Platforms that provide a similar solution to React Native include Apache Cordova and Google's Flutter. Google's Flutter does much of the same things as React Native but since it is in its infancy the stability of the product is not as high as React Native (References #9). Apache Cordova uses a different strategy for making native code. It wraps web apps in a browser container and packages it as a mobile app(References #15). This makes the performance much less efficient.

2.4.2 Relevant Literature

All relevant literature are articles written by seasoned developers all across the industry.

Switching over to React Native will save companies immense resources by compiling multiple teams and applications to a singular one. “ If Valtech and other developers would begin to use React Native, they would only need one team that can create all three services with more or less the same code”(Reference 18).

There are multiple other possible parties, libraries that do something similar, “All four technologies allow you to build real native mobile apps for both iOS and Android” (Reference #16), but React Native has the largest community.

Other than these scholarly articles, a main source of information would be the documentation page of the appropriate technologies.

In addition to documentation, we will be reading various information from places such as GitHub, StackOverflow, and other online platforms. This will help us understand what the current techniques and approaches are for solving issues that we come across.

2.5 Proposed Design

We have a unique project because the application we are building already exists. Our job is to remake the application using a new framework. Thus, the design for most of our application (especially the UI design) is already complete. Our project consists of creating the new react native code base and recreating the components found in the Buildertrend application. We are not responsible for creating any backend code. This leaves the underlying architecture design to us along with design aspects relating to communicating with Buildertrend's API.

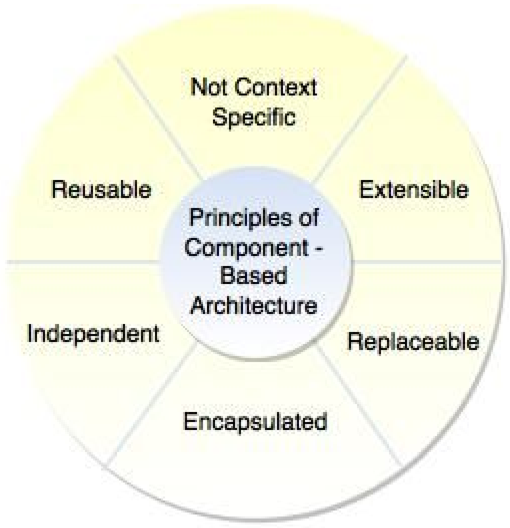

Figure 1: Principles of Component Based Architecture

Our strategy for the architecture includes using a component-based architecture and using a central store to save the application state. Also, instantiating an interface for API calls. A component-based architecture allows the pieces of functionality to be replaceable, independent, reusable, extensible, and not context specific. This will make our code DRY and easily modifiable. One of the motivations for this project, to begin with, is having a code base that is used for the development of both ios and android. Having a component-based architecture will ensure the modifiability of a unified code base.

The other strategy for the application design is using a central store to store global state variables. User data used across multiple components can be shared in the central store. Each component would not need to make requests for the same data which cut down on redundant API calls. Also, when a component needs to make an API call, the values of the call may depend on a value set by a different component. Instead of relating these two components, the value can be accessed from the central store reducing cohesion and satisfying the properties of the component architecture.

The API communication interface will provide all functionality for making calls to the API. This design construction reduces repeated code by providing functions for the API routes with the ability for components to pass in their own parameters again making the code base DRY. This component will have access to the central store for any parameters that come from the state of the application.

In the functional requirements section, we detail all aspects of the application that needs to be implemented. Many of the modules can be instantiated with the same components. For example, the search feature is common for many modules within the app. We will create one search component that can be used in each of the modules. The functionality of the component will replicate that of the original application.

2.6 Technology Considerations

- TypeScript - We picked TypeScript as the language used for development instead of JavaScript. One reason we picked TypeScript is that it is a typed language which will help us avoid tricky bugs; it also has other feature languages that don't need a transpiler like JavaScript(besides the TypeScript transpiler itself). One weakness of TypeScript is that there might be some time spent in learning the language instead of actual development.

- NPM - we are using npm and yarn as our package managers. We use yarn primarily for package managing because it has a lock file that will help us manage dependencies across all our systems. npm does offer a lockfile now but we decided to go with yarn because of its reputation for speed.

- Postman - we are using Postman to get the HTTP request used by the existing mobile application to get HTTP routes to the server. Some of the alternatives were Charles and Fiddler, but we went with Postman because most of the team has experience with Postman.

- React Native - The client requested that we use React Native. Alternatives to this are Flutter, Iconic, and Xamarin. The client already chose the technology we should use so it wasn't considered by us. One of the strengths is the popularity and community.

- Expo - Expo is a tool for mobile development. React Native does not have a development tool like Expo available, which requires us to use both for efficient development.

- React-Devtools Debugger - We use this for debugging React Components. Alternatives include the IDE debugger and the mobile's inspector. We chose React-Devtools because it was made specifically for React/React Native so it is geared towards what we are developing and has some specific features to React Native.

- Redux - We are using redux for state management within the mobile application. Other alternatives were flux and mobx. We chose redux because it has more community support and user-friendly. Flux offered almost no advantages. Some of our team members were also acquainted with redux so we picked it over mobx. One of the weaknesses/trade-offs with redux is the boilerplate that has to be done before the project can be started.

- React Router - We are using react router for navigation within the application. One of the strengths of this is the efficient management of navigation which doesn't just help us do the work of navigation but also in optimal time. Some alternatives were React navigation, and React Native navigation. React navigation seemed like a more immature library compared to the other two with some API constraints. React navigation was a suitable alternative but is more difficult to set up and didn't seem to offer any new features that aren't in React Router.

- NPM - we are using npm and yarn as our package managers. We use yarn primarily for package managing because it has a lock file that will help us manage dependencies across all our systems. npm does offer a lockfile now but we decided to go with yarn because of its reputation for speed.

- GitLab - We considered using another remote source control distribution, namely, GitHub because most of our team was familiar with it, but we decided to go with GitLab because it was already set up for us, it offered better issue tracking and easier code review and enforcement than GitHub.

- Linting - We are using TSLint to ensure code consistency in our project. Other alternatives include JSLint and ESLint. We picked TSLint because it is made specifically for TypeScript while the rest of them are geared towards JavaScript.

2.7 Safety Considerations

There are very little safety considerations in our project. Due to the nature of our application being used in construction areas, we have to make sure that the product runs reliably and efficiently to ensure that they are not waiting on the frontend of the application.

We also have to make sure our application doesn't cause unnecessary distractions, such as flashing lights or loud noises. The safety of the user base is important to Buildertrend and therefore is important to us as well.

2.8 Task Approach

With all six people on the team having different class schedules, it is necessary for us to have a reliable tracking system to ensure that work gets done efficiently without overlap.

The main immediate method we use to help keep track of tasks is the ticketing system that's built into the GitLab repository that we are using for the project. This is more of an agile method.

The method we are using to keep track of the overall plan is a Gantt Chart. We have all the major milestones laid out for the year so we can see that we are on track to finish.

To see figure, check *Figure 2: Semester 1 Gantt Chart* and *Figure 3: Semester 2 Gantt Chart*.

2.9 Possible Risks And Risk Management

Title: Lacking Understanding of React Native Framework

Strategy: Avoid

Premise: Coming into this project, most of the team members have not used the react native framework. Lacking understanding of React may cause slips in schedule and loss in quality as members take time to grasp react. Estimating the required amount of time for tasks will be difficult because we have no understanding of our efficiency with React.

Action Plan: Getting as much experience with the framework will give us the best results if we are able to make more informed guesses and utilize aspects of React that improve quality. We plan on doing tutorials and reading documentation to understand React better.

Title: Relying on Phones

Strategy: Mitigate

Premise: React native compiles native code for android and ios phones. In order to run the application, team members need to use their phones. Relying on phones in the workflow introduces risks such as not having a working phone, not having a charged phone and compatibility between the phone and the application. It is possible to create virtual devices to run the application but this may prove to be difficult.

Action Plan: We will work in groups so that phones can be shared if need be. There will probably be at least one person with a working phone. Also, we will look into the option of having virtual devices and the feasibility of this option.

Title: Malformed HTTP Requests

Strategy: Mitigate

Premise: The application we are developing uses the Buildertrend backend functionality through their API. However, we do not have any documentation on the API and therefore must intercept requests on their app to find out how it is used. Not having access to the API documentation will cause a lack of understanding and therefore may cause incorrect or malformed HTTP requests.

Action Plan: Having one person dig deep into the HTTP calls and having a map of what each does will provide other team members with the knowledge to use them correctly. When Buildertrend's API doesn't return what we expected, we can rely on the person who researched it more closely. Also, we can use our contact at Buildertrend to ask questions about what might be going wrong.

2.10 Project Proposed Milestones and Evaluation Criteria

Some key milestones involve an initial prototype, a structured prototype and a finished product. The initial prototype will include having an app that works on Android and iOS. This will be the frontend and the skeleton of the product using dummy information. We will test this by using phone simulators as well as an Expo app that allows us to run the React Native code on our devices. We will use manual tests as well as testing frameworks such as jest or enzyme.

The structured prototype will be an improved version of our initial prototype with 75% of the capabilities required for our product. One of these key capabilities will be pulling correct information from the Buildertrend servers. We can test this by comparing the information on our newly built app to the old app that they use. We have been given a dummy account by Buildertrend so we can use this account to test our app. The frontend should already be working and displaying dummy information correctly at this point so we will just need to test the new information being provided. This will include manual tests as well as generated test cases that can compare the expected information with the information being pulled in.

The last milestone is the finished product. This will be when our app is displaying all the correct information as well as having our frontend completed. Documentation for our project will also be completed and ready to turn over to the client. The app should work on both an Android and an iPhone. We can test our finished products on both devices as our team has both available to them. We can compare this to the old app and to expected values as we run manual tests to make sure this is working correctly. We will also be sure to test with generated test cases as well as testing old tests to make sure that we did not defect from our code as we were improving upon it.

2.11 Project Tracking Procedures

To track progress our group has decided to use the GitLab issue tracker. We will use this to create tickets of key issues that need to be resolved as well as tasks that need to be completed. As we complete the tasks we will take note of these and continue to follow our proposed timeline.

The tasks will be created to follow our schedule closely and work towards meeting our milestones and eventually our deliverables. The issues that will be created will be brought up as problems arise in our code and will be handled and dealt with accordingly. We will continue to compare our progress with our timeline to stay on course and have our product finished appropriately.

2.12 Expected Results and Validation

The desired outcome for the end of this project is to have a smooth application running on both Android and IOS that hits on all of the functional and nonfunctional requirements. We will ensure this desired outcome happens by staying on track with our Gantt chart and fulfilling every task that comes this way.

A common stumbling block many teams encounter is unsustainable code. If the repository gets populated with subpar code, it hinders the scalability of the project. Since this project is meaningful and actually needs to be maintainable, our team decided to have a strict code review policy that makes it so at least three other developers have to perform a code review and approve the code.

With the intense code review in place, it helps maintain scalable code, which helps ensures that our outcome will be scalable.

We will also have a strong testing procedure that includes manual testing on our phones using Expo and testing frameworks *Jest* and *Enzyme*.

2.13 Test Plan

Testing is an important part of developing an application. We plan to use regression testing throughout the development cycle. The regression tests will ensure that any newly implemented code doesn't break previous functionality. This will be done by writing test statements for desired functionality.

In order to do this, we will be using a javascript testing library called Jest. In addition to the regression testing, we will be using functional testing. This will ensure that each component is outputting the necessary items.

Enzyme will be used regressively to make sure that every tag is used appropriately for every component. This will ensure that every component renders everything that it is supposed to. This will, in turn, check the functionality of the separate components within our application.

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

Our project timeline has been broken into three phases. These phases are based off our milestones which are an initial prototype, a structured prototype, and a final product. These phases take up a duration of the Fall 2018 semester as well as the Spring 2019 semester. The phases involve the implementation of that prototype, testing, and other steps that our team will take to complete the finished product.

The phases are broken up into two to three-month segments with the structured prototype being the shortest one. This structured prototype will be built off of the initial prototype and eventually, the finished product will be completing the structured prototype to be packaged and ready to present to the client. We believe creating the basic prototype will take longer as we work to create components to build our app off of. The second phase will be applying these components to different capabilities of the app. The final phase will involve documenting our code as well as testing and preparing our software for the client. Please reference our Gantt charts represented in figure 1 and 2 to view the timeline described below.

Phase 1: Initial Prototype (9/1-10/31)

- Building the skeleton code
- Basic app functionality
    - Basic components
    - Homepage
    - Login page
- Having the app work on iOS and Android
- Testing of app functionality

Phase 2: Structured Prototype (10/31-11/30 & 1/14-2/8)

- Increased app functionality
- Successful compatibility with BT services
- Testing
    - App functionality
    - Compatibility with BT services
    - Regression testing

Phase 3: Final Product (2/9-4/30)

- Finalize app functionality

- Deliverables completed
    - Documentation
- Functional and Non-functional requirements completed
- Testing
    - Functionality
    - Regression testing
    - Requirements
- Packaged and ready to deliver

3.2 Feasibility Assessment

The application we create should meet the standard of an industry leading company. Although there are challenges that lay in the way, the application will meet the functional requirements that were asked of us. This React Native app will solve the companies problems of maintaining two applications and saving them money and time.

Going into the project, it is clear that there are some strengths in our solution that may help us succeed. For example, almost everyone in the group has experience in frontend development. TypeScript is very similar to JavaScript, so there should not be many roadblocks there. React is also a tool that most of our team commonly uses and will be essential to completing this task. Another advantage we have is that a few of our members have experience working with Buildertrend. Frank, Walter, and Kyle are past interns, so communication with the client will be effective, and we already have an idea of how the desired product can be created. Our greatest strength may be that our team has experience working together. We have worked on projects in the past, and we seem to have a great work dynamic.

There are several weaknesses in this proposed solution. Since every member has other responsibilities, each developer has to juggle school, their current jobs, and much more. Another weakness is how it can be difficult for us to have consistent times with our client and advisor that every developer can attend. All of us have different schedules which can also slow down the development process in general. The only way to complete the project thoroughly and on time will be to keep each other in check.

Another weakness in our proposed solution comes from the fact that Buildertrend, our client, will not be giving us access to the backend API. This means that our process will be slowed down in order to find a way to monitor API calls. Another roadblock is the fact that our client is a long distance from where we are located. If we have a question or something that direly needs to be discussed, this may be a problem.

A final weakness of this proposed solution is the time we have to complete this project. The project that we have is literally the entire Buildertrend mobile application. Buildertrend has been working on this project with dozens of developers for years. It will be a challenge to complete every single component that they have built out for the previous native apps. We would like to complete all the components, but it will require a lot of effort from all members. Ideally, we would like there to be little work left for Buildertrend to deploy our project once we are done.

Overall, the strengths of our proposed solution will outweigh the weaknesses. Where we will have to spend the time to learn new technologies, we also all have the basic skills to program efficiently with frontend tools. We all have different skill sets, but our communication skills allow us to overcome individual weaknesses that we face. There is a lot to do, but with an effective schedule, members keeping others in check, and clear communication with all parties, we will have a productive and rewarding development process.

3.3 Personnel Effort Requirements

This section is a work in progress. The estimated hours are generally how long we expect tasks to take. Since we have not completely most of the tasks, it actual hours could change.

Below lies Table 1, that contains all of the major tasks, with description and estimated hours necessary for each task.

Task	**Description**	**Estimated Hours**
Understand project	Contact with Client, discuss expectations, and mess around with demo application	20 hours
Research technologies to use	Researching possible software frameworks and packages and weight the pros and cons for each one	30 hours
Create initial application	Create skeleton of React Native App, Inject Redux into project, install dependencies, and everything else we agreed on from researching technologies	15 hours
Set up environment	Make sure that the environment is set up for every device for each team member	10 hours
Research on software Architecture	Before we get too far into the project, settling on an architecture to model the files is crucial	10 hours
Create Scripts	Scripts will help speed up development process for the future. Scripts to start application and test	30 hours
Discover API	Since we are not given the API for the backend calls, we have to use another tool,	50 hours

	Postman, to track every API call.	
List Components to create	Go through the old application and list out every possible component and where it is used.	20 hours
Create Menu and framework	Having a main infrastructure to help the user move around the application is necessary before we get into the big parts	30 hours
Individual Development	Split up components for each member	150 hours
Weekly Reports	Keep track of past week's work and deploy them into a report every week	30 hours
Write Tests	Create tests that assure correct functionality and display	50 hours
Code Review	Not just developing, but reading other people's code before their work is officially approved into the repository.	100 hours
Project Plan	Come together to work on the massive project plan report	30 hours
Document Software	Team members need to document all software code, architectures used, and design patterns used.	50 hours
Test requirements	Test requirements we laid out and address them if need be	20 hours

Table 1: Major Tasks

3.4 Other Resource Requirements

Hardware is a resource outside of finance that is required to develop this application. Every team member has to have a computing device that can have access to the internet and have a text editor. Ideally, the computer will have enough processing power to run an IDE that is capable of installing NPM packages and running the application.

Lastly, having a smartphone handy will help with the manual testing of this application. Allowing each developer to complete regression tests, generated tests, and manual tests.

3.5 Financial Requirements

The tools that we are using are all currently open-source. React Native and Redux were created by Facebook and can be used freely by the community. There are also tools such as editors, learning tools, and example projects that are also free to use. The majority of the software is always free for personal use, but if the creators of a tool decide to make it cost for a business such as Buildertrend to use, we will expect Buildertrend to cover these costs. There is no exact expected cost, but we are hoping the cost of creating the software will only cost us time. We are early in the development process, so we can update the cost as we go if we run into problems. Normally, there would be the cost of paying the developers, but we are working for free.

4 Closure Materials

4.1 Conclusion

Updating an application from pre-existing legacy code is a huge part of software development in today's technology industry. Many times companies find themselves wasting resources by having developers rewrite or remake existing applications on separate platforms.

Our project aims to transition two separate native applications into a common React Native application. This will reduce the cost of upkeep and allow a fluid user experience across all platforms, saving Buildertrend both time and money.

In order to make this transition as smooth as possible, it will take the collaboration of all six team members as well continuous communication with our client. Along the way our academic advisor Mai Zheng will be there as a support system for any questions or concerns that may come up along the way.

As outlined in this project plan, we will continue to work towards transitioning this industry leading software to a more up to date technology. By doing this we will not only improve the efficiency of Buildertrend's development team, but we will also create a refined product that extends to home builders around the world. This product will benefit the likes of Buildertrend, the builders, and their clients alike.

4.2 References

Technology:

1) https://www.ieee.org/about/compliance.html
2) https://jestjs.io/
3) https://airbnb.io/enzyme/docs/api/
4) https://redux.js.org/
5) https://medium.freecodecamp.org/8-key-react-component-decisions-cc965db11594
6) https://medium.com/@dbow1234/component-style-b2b8be6931d3

7) https://buildertrend.com/
8) https://www.typescriptlang.org/
9) https://facebook.github.io/react-native/
10) https://expo.io/
11) https://www.npmjs.com/package/react-devtools
12) https://reacttraining.com/react-router/core/guides/philosophy
13) https://yarnpkg.com/en/
14) https://www.getpostman.com/
15) https://cordova.apache.org/
16) https://flutter.io/
17) https://academind.com/learn/flutter/react-native-vs-flutter-vs-ionic-vs-nativescript-vs-pwa/
18) http://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02

4.3 Appendices

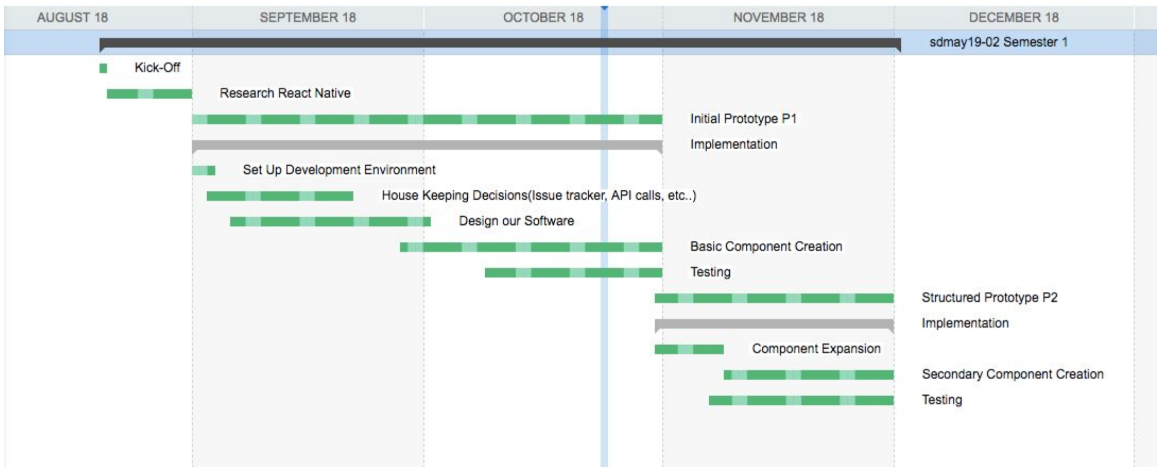

Figure 2: Semester 1 Gantt Chart

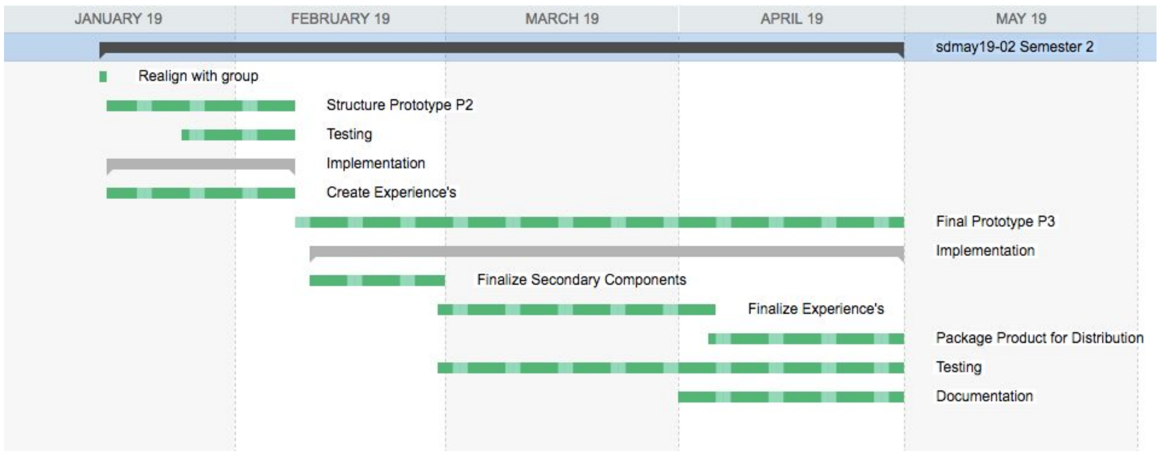

Figure 3: Semester 2 Gantt Chart