DocuSign Integration

DESIGN DOCUMENT

Team 28

Client: BuilderTrend

Advisor: Dr. Ashraf Gaffar

Justin Rule, Carson Meyer, Joe Slater, Ale Groe, Alan Zapinski, Conley McKown, Jack Goldsworth

sdmay22-28@iastate.edu

https://sdmay22-28.sd.ece.iastate.edu/

Revised: 12/5/2021/Version 2

Executive Summary

Development Standards & Practices Used

IEEE-29119: We will write Unit Tests for our web application to ensure functionality in each separate piece.

IEEE-12207: We will follow the standard software lifecycle process.

IEEE-7.8: Code of ethics: We will follow ethical development practices.

IEEE-1363: Encryption. We will follow standard hashing and encryption policies to store user data, namely passwords and logins.

Summary of Requirements

- We must avoid the DocuSign API rate limit (**constraint**)
- Our web application must demonstrate the main use cases for DocuSign in BuilderTrend:
    - Homeowner + Contractor signatures
    - Multiple contractor signatures
    - Homeowner + Builder signatures
- Our web application must use modern Javascript frameworks
- Our application must be encrypted and secure when dealing with contracts (**constraint**)
- Thorough documentation about DocuSign's API + use cases
- Our documentation must be organized and readily accessible to BuilderTrend developers
- There must not be significant delay when sending documents in our web app
- Our application must be developed using Agile methodologies

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- ComS 227, 228, & 327 - General coding skills and good practices
- ComS 309 - Learning team-based development
- ComS 319 - Interface design
- SE 329 - Project management (Especially important)
- ComS 363 - Database design

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- Jack: I learned how to take application direction from a client and then communicate those requirements to a team and other involved parties.
- Joe:
- Justin
- Ale
- Carson
- Alan
- Conley: I learned how to work with a client without much contact. Taking a list of requirements and writing a project plan without being guided through it.

Table of Contents

List of figures/tables

1 Team

1.1 Team Members

Jack Goldsworth, Conley McKown, Ale Groe, Joe Slater, Justin Rule, Alan Zapinski

1.2 Required Skill Sets for Your Project

- Concise and good communication
- Front-end software development
- Back-end software development

1.3 Skill Sets covered by the Team

- Concise and good communication - Ale, Carson
- Front-end software development - Jack, Justin, Conley, Joe
- Back-end software development - Jack, Justin, Ale, Joe, Alan, Conley

1.4 Project Management Style Adopted by the team

Agile methodologies and bi-weekly sprints.

1.5 Initial Project Management Roles

- Jack - Research appropriate frameworks and languages to use.
- Justin - Coordinate meetings with TA.
- Joe - Outline front-end goals.
- Ale - Research other products to use to implement this idea.
- Conley - Outline back-end goals.
- Alan - Project timeline coordinator.
- Carson - Client Communication.

2 Introduction

2.1 Problem Statement

Buildertrend is a company that creates and maintains a management software for different types of construction projects, but especially for the constructions of new homes. Their software grants the people involved better communication and coordination, resulting in the job being done better and faster. One important part of the process is contract signing, and being able to securely manage contracts between clients is what we seek to provide Buildertrend through the implementation of DocuSign into their product. DocuSign is a third party company that provides the ability to securely sign documents, and we have been tasked with researching DocuSign, creating useful documentation on how to best build it into their software, and creating a 'proof of concept' web application demonstrating the usefulness of our documentation.

2.2 Requirements & Constraints

List all requirements for your project . This includes functional requirements (specification), resource requirements, qualitative aesthetics requirements, economic/market requirements, environmental requirements, UI requirements, and any others relevant to your project. When a requirement is also a quantitative constraint, either separate it into a list of constraints, or annotate at the end of requirement as "**(constraint)**". Other requirements can be a single list or can be broken out into multiple lists based on the category.

- We must avoid the DocuSign API rate limit (**constraint**)
- Our documentation must allow for easier implementation of the main use cases presented by Buildertrend:
  - Homeowner + Contractor signatures
  - Multiple contractor signatures
  - Homeowner + Builder signatures
- Our web application must use a modern Javascript framework
- Our application must be encrypted and secure when dealing with contracts (**constraint**)
- Our web application must be able to demonstrate the ease of implementation provided by our documentation
- Thorough documentation about DocuSign's API + use cases
- There must not be significant delay when sending documents in our web app
- Our application must be developed using Agile methodologies

2.3 Engineering Standards

- IEEE-29119: We will write Unit Tests for our web application to ensure functionality in each separate piece
- IEEE-12207: We will follow the standard software lifecycle process
- IEEE-7.8: Code of ethics: We will follow ethical development practices
- IEEE-1363: Encryption. We will follow standard hashing and encryption policies to store user data, namely passwords and logins.

2.4 Intended Users and Uses

Our project will be presented in two main deliverables. First, we will be creating documentation for developers at Buildertrend to use about DocuSign's API based on the use cases presented to us by them. This documentation should be thorough and make implementation into their preexisting code smoother. Second, we will create a web application by following our documentation to demonstrate its usefulness. Along the way we will be updating the documentation as we run into roadblocks and put solutions in place so as to make some common pitfalls easier to avoid when following our documentation. This is not only useful in proving the validity of our documentation, but also allows management and other stakeholders to see the success of our implementation and how it can be useful for Buildertrend.

2 Project Plan

2.1 Project Management/Tracking Procedures

We are following the agile model for our project for a few reasons. The first one is that the scope of our project is not large enough to justify needing to use the waterfall model. Our lifetime of the project that we are working on, and the open-endedness of the project that Buildtrend has given us means that we can use the agile project management style. Another reason that the agile cycle will work for this project is that it will allow us to quickly adapt to any changes that Buildertrend would like us to make. For instance, if we are given access to Buildertrend's code, the project's technologies might need to change to accommodate that.

This project will be using GitLab's task management to keep track of the progress that team members are making. GitLab will also allow the tasks to be assigned to working code branches, commit history of those branches, and long term comments that shouldn't be stored in any communication apps. We may also use Trello so that we stay organized using boards and track overall project statuses.

2.2 Task Decomposition

Task 1 - Discovery and Documentation - Research and analyze how Docusign's API can best be utilized to implement Buildertrend's specific use cases. Compile relevant information on how to implement the API with Buildertrend's technologies. This task is concurrent in execution with task 2 and 3.

- Learn about Docusign's API
- Legality of use with Docusign
- Compiling information learned into implementation documentation
- Study implementation demos from Docusign
- Learn Docusign basics and terminology
- Document and address shortfalls in Docusign API

Task 2 - Demonstration - Interact with API to produce better documentation, as well as pinpoint and develop solutions for any avoidable pitfalls in the implementation

- Make demo with the API to demonstrate basic use cases
- Document and address shortfalls in Docusign API

Task 3 - Minimum Viable Product - Utilize previous research, documentation, and experience to present a minimum viable product that addresses Buildertrend's specified requirements

- Confirm all requirements with Buildertrend

- Demo app with Buildertrend all use cases

Task 4: Stretch Goals - These include the possibility of Docusign implementation with Buildertrend codebase, mobile signature app, or local document storage management with template documents.

- Further subtasks for this have yet to be defined as these are stretch goals.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones

M1: Complete all initial class course work, including the project/testing/design/professionalism plans, and any other in-class work.

M2: Complete research into Docusign and it's documentation in order to determine its capabilities.

M3: We integrate the DocuSign API into a test application for an initial proof of concept. This will include a few basic use cases.

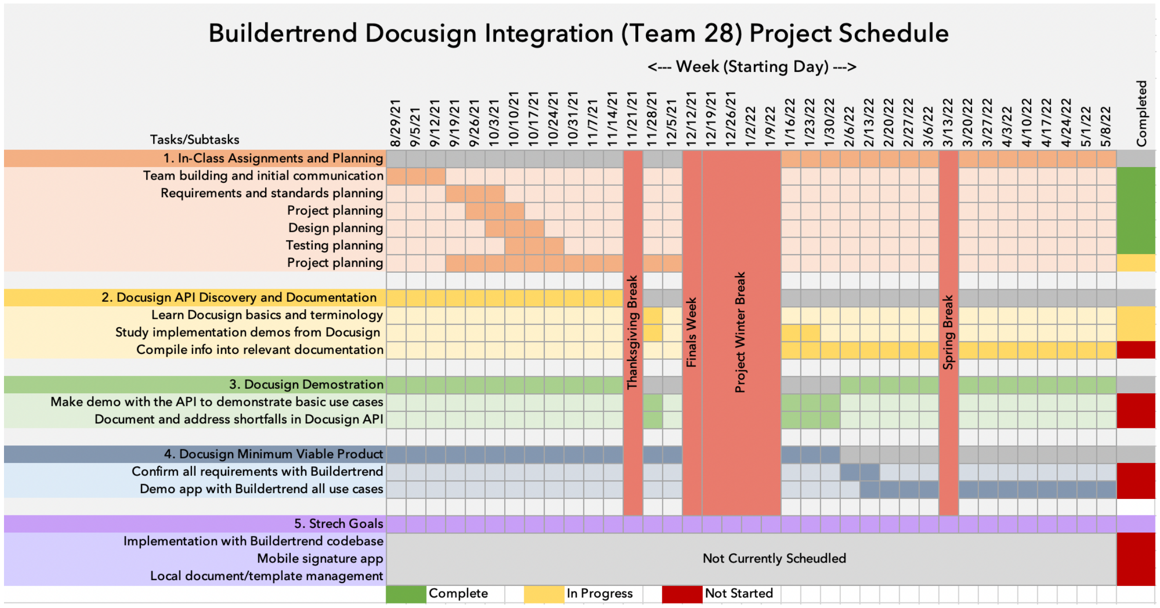
M4: Integrate all use cases from Buildertrend and any additional ones that we believe are critical to software functionality.

M5: We pick a stretch goal option, and complete it.

Evaluation

Our Team will evaluate the success of these milestones with the completion of the subtasks identified underneath each task.

2.4 Project Timeline/Schedule (Figure 1: Project Schedule)



2.5 Risks And Risk Management/Mitigation

Risks: These are the risks that we have identified so far. These are ranked low to high. Anything with a high risk we will develop a risk management plan for it.

Task 1 - Discovery and Documentation -

- Can't use Docusign API because of the legality of our use. - Medium risk

Task 2 - Demonstration -

- Our basic demo reveals serious problems with the way that Docusign works. - Low risk

Task 3 - Minimum Viable Product -

- Advanced demo and minimum viable product show that Docusign won't work with Buildertrends use cases. - Medium risk
- Our proof of concept application cannot demonstrate complex use cases. - Low risk
- Buildertrend changes the scope of the project and we have to spend more time back on task 1. - Low risk

Task 4: Stretch Goals -

- There are no risks associated with this right now as we are currently not looking in the scope of this.

Non-task related Risks:

- Data retention doesn't match what Buildtrend wants. - Low risk
- Contract data is compromised through DocuSign's platform. - Low risk
- Attackers spoof our application to expose user credentials and view confidential information - Medium risk

2.6 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in total number of person-hours required to perform the task.

Table 1: Task estimate

Task #	Hrs Estimate	Current Hrs Spent
Task 1 - Discovery and Documentation	75 hrs	180 hrs
Task 2 -	135 hrs	15 hrs

Demonstration		
Task 3 - Minimum Viable Product	375 hrs	0 hrs
Task 4: Stretch Goals	To be determined	N/A

A lot of our time estimations came from prior working experience on software projects. All of our team members have completed COMS 309 which provided us with experience planning and executing a semester long project. Many of us have also completed software engineering internships which were very helpful in providing experience regarding timeline estimations. Additionally, we consulted a few different websites, listed below this paragraph, to try and make these estimations as accurate as possible. As we progress through the semester, we can easily refine these guidelines. These times can be mapped to the corresponding dates in the Gantt chart/schedule above.

Sources:

https://medium.com/globalluxsoft/time-estimation-in-software-development-a4a495c8eb6c, https://winatalent.com/blog/2020/02/time-estimation-in-software-development/

2.7 Other Resource Requirements

Because of the portability of the Docusign API, and the frameworks and languages we are using right now our project will not need any extensive materials. There might be a possibility in the future that we need a web server for our application, but the development should be able to take place on our laptops.

3 Design

3.1 Design Context

3.1.1 Broader Context

BuilderTrend provides construction project management software primarily used by builders, homeowners, and subcontractors. Our DoucSign solution will make signing agreements easier, more scalable, more virtual, and more searchable.

Table 2: Broader Context

Area	**Description**	**Examples**
Public health, safety, and welfare	This project allows for all parties to sign their legal documents electronically	Reduction in Covid-19 exposure.
Global, cultural, and social	It will help communication between builders, homeowners, and contractors as well as keep everyone accountable for	Implementation would reduce face to face interaction.

	completing their tasks in a timely manner.	
Environmental	Reduce material usage as all documents would now be stored electronically.	Decreased usage of recyclable materials.
Economic	This project will be viable for BuilderTrend and will cut down on their material costs.	Product will remain affordable for BuilderTrend and allow for more accessible, cost effective use of their product.

3.1.2 User Needs

We have identified three groups of users, all with slightly different requirements:

Builders: Builders need a way to manage, sign, and review contracts between themselves, the homeowner, and contractors. This is because a builder should facilitate the relationship between a homeowner and a contractor to complete a construction project.

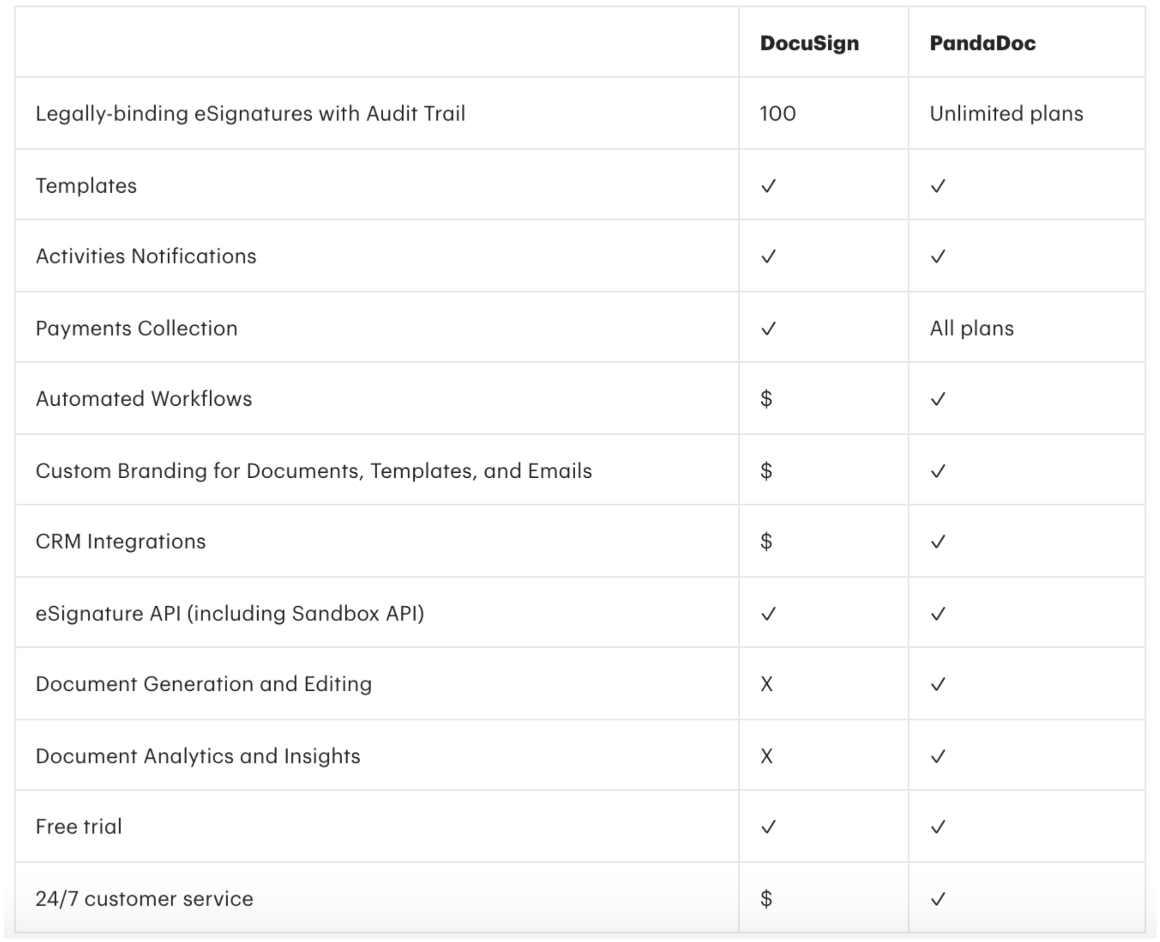
Homeowners: Homeowners need a way to review and sign contracts related to the construction of their home, as well as viewing these documents at any time. This is because a homeowner will need to make decisions regarding their new home, and those decisions need to be formally agreed upon via a contract that can be accessed at any time.

Contractors: Contractors need a way to create, share, and sign documents related to their work on a house or construction project. This is because the contractor will complete specific parts of a project, and they need to receive payment according to a contract.

3.1.3 Prior Work/Solutions

For our project, there is not a lot of background/literature other than understanding what BuilderTrend does and who will be using our product. This information can be found in other sections. However, since we will be relying on the DocuSign API for most of our project (They have multiple APIs, but we will likely be using their eSignature API since we need to capture actual signatures). There are many, many alternatives to DocuSign, but many of them are targeted towards smaller businesses. A few examples of these are HelloSign and PandaDoc HelloSign's specific advantages are it is easier to set up, it offers more features in its free plan than DocuSign, and supports in-person signatures.

DocuSign's advantage is it is targeted for large commercial companies who need to process many signatures and documents (Similar to what BuilderTrend needs). DocuSign also offers a mobile app in its paid tiers. Below is a feature comparison chart taken from competitor PandaDoc's website.



	DocuSign	PandaDoc
Legally-binding eSignatures with Audit Trail	100	Unlimited plans
Templates	✓	✓
Activities Notifications	✓	✓
Payments Collection	✓	All plans
Automated Workflows	$	✓
Custom Branding for Documents, Templates, and Emails	$	✓
CRM Integrations	$	✓
eSignature API (including Sandbox API)	✓	✓
Document Generation and Editing	X	✓
Document Analytics and Insights	X	✓
Free trial	✓	✓
24/7 customer service	$	✓

Figure 2: DocuSign vs PandaDoc

It is also important to note that BuilderTrend explicitly stated that they want us to use DocuSign, so we will not be using a competitor. As far as following previous work, we will need to integrate our code into their pre-existing platform. We do not have access to their code base right now, so we do not know exactly what that will look like.

3.1.4 Technical Complexity

The ability to account for multiple signatures in a project has added complexity to our design.

There are three possible ways that the DocuSign API can be used: Single signer, multiple required signatures, and one signature required but multiple people can sign for it.

Another big portion of our project is the documentation and knowledge transfer aspect. This means that there is infact design complexity within our documentation, due to the fact that it needs to be written so that an engineer can replicate our results as efficiently as possible (User manuals, JSDoc, DocuSign API Endpoints).

We must create a full-stack web application, which provides complexity in requiring the frontend, backend, database, and third party API to seamlessly communicate.

3.2 Design Exploration

3.2.1 Design Decisions

List key design decisions (at least three) that you have made or will need to make in relation to your proposed solution. These can include, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, features, etc. (tech stack, system design, features)

We made significant decisions regarding our implementation and technology:

- **Tech Stack:** We decided on a full Javascript stack, with a MySQL database and React frontend. This is explained in more detail in section 3.4: Technology Considerations.
- **System Design:** We needed to design a system that would allow users to interact with a web application, while under the hood, we can interact with all facets of the DocuSign API. Thus, the obvious choice was a client/server relationship where the server acts as a REST API that the client can call. With these applications running separately, the server can make calls to the DocuSign API instead of funneling this traffic through the client. This allows for much more customization and logic to be implemented.
- **Features:** While BuilderTrend provided a list of features to us, they mentioned that it was fairly open-ended. We had the ability to expand on features as we see fit - as long as it provided useful functionality. Thus, we decided to emphasize the following features:
    - The ability to require signatures from multiple parties on a document
    - The ability for both signers and signees to see the status of their documents at any time
    - The ability to update a document before it has been signed

3.2.2 Ideation

There were lots of options for our technology stack, including:

- React JS frontend, no backend (frontend calls DocuSign API directly)
- React JS frontend, C# backend
- Angular frontend, C# backend
- **React JS frontend, Node JS backend**
- No usable frontend, we just provide a REST API to connect to DocuSign

We considered the pros and cons of each option, heavily weighing the interests of our client. In the end, this application is most useful as a proof of concept website with a usable React frontend, and Node JS backend.

3.2.3 Decision-Making and Trade-Off

Demonstrate the process you used to identify the pros and cons or trade-offs between each of your ideated options. You may wish you include a weighted decision matrix or other relevant tool. Describe the option you chose and why you chose it.

As a team, we discussed the pros and cons of each technology option. Below is the summary:

- React JS frontend, no backend
    - Pros: We are all familiar with javascript, not making a backend is simpler
    - Cons: No backend makes data storage very difficult, and our solution is less customizable
- React js frontend, C# backend
    - Pros: Many of us are familiar with React, the backend gives flexibility, BuilderTrend uses C# as a backend language
    - Cons: We are unfamiliar with C#, and not sure if DocuSign has a C# SDK
- Angular frontend, C# backend
    - Pros: Angular is a powerful Javascript framework, we are familiar with Javascript
    - Cons: We are unfamiliar with C#, and not sure if DocuSign has a C# SDK
- **React JS frontend, Node JS backend**
    - Pros: Many are familiar with React and Javascript, there is a well-documented official Node SDK for DocuSign, we can keep both layers in the same language
    - Cons: BuilderTrend does not use a Node JS backend
- No frontend, Node JS backend
    - Pros: We can focus more energy into the DocuSign functionality
    - Cons: Very difficult to demo, not very user-friendly

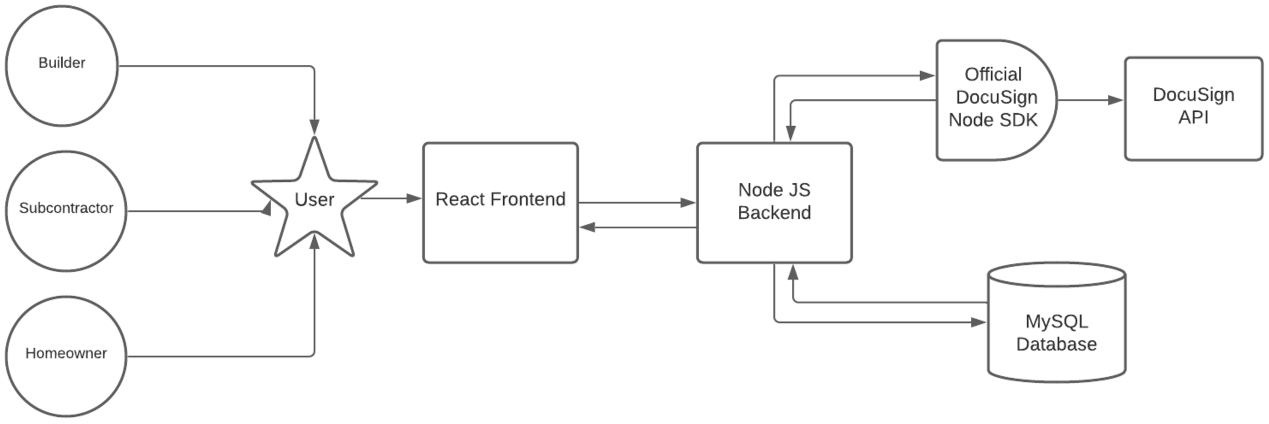
We chose a React JS frontend and Node JS backend.

See section 3.4 for more detailed technology considerations.

3.3 Proposed Design

3.3.1 Design Visual and Description

This is an architectural diagram showing the major components to our application. *Figure 3: Design*



Our design (shown above) uses a layered software architecture to deliver a full-stack web application that is user friendly while still delivering on all major requirements. End users will only interact with the React JS frontend application - this is accessed in a web browser. All user requests are then passed to the Node JS backend, where they are handled. Depending on the request, our implementation can either make a call to the DocuSign API or retrieve data from our local MySQL database.

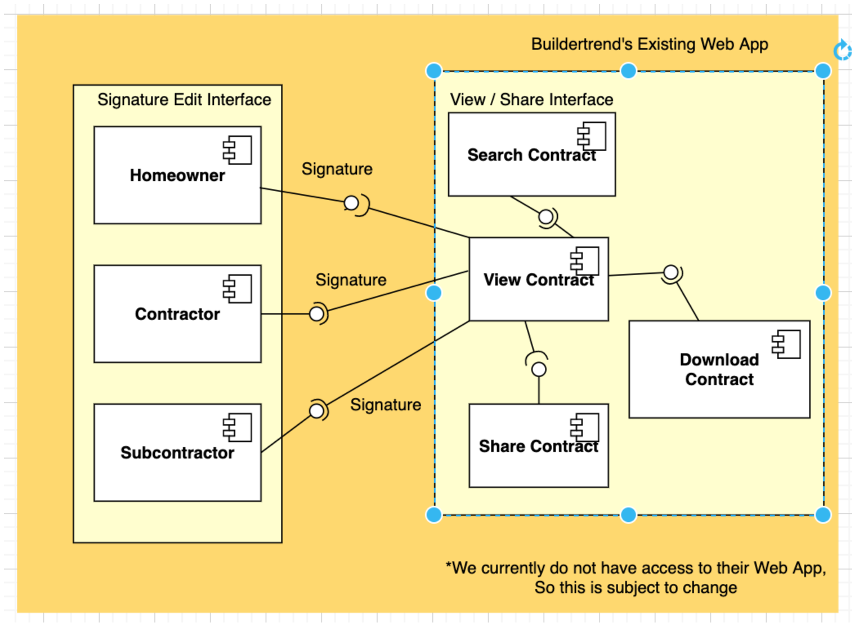


Figure 4: Component Diagram

Above is a component diagram that shows the relationship between the different parts of our design that users will be interacting with.

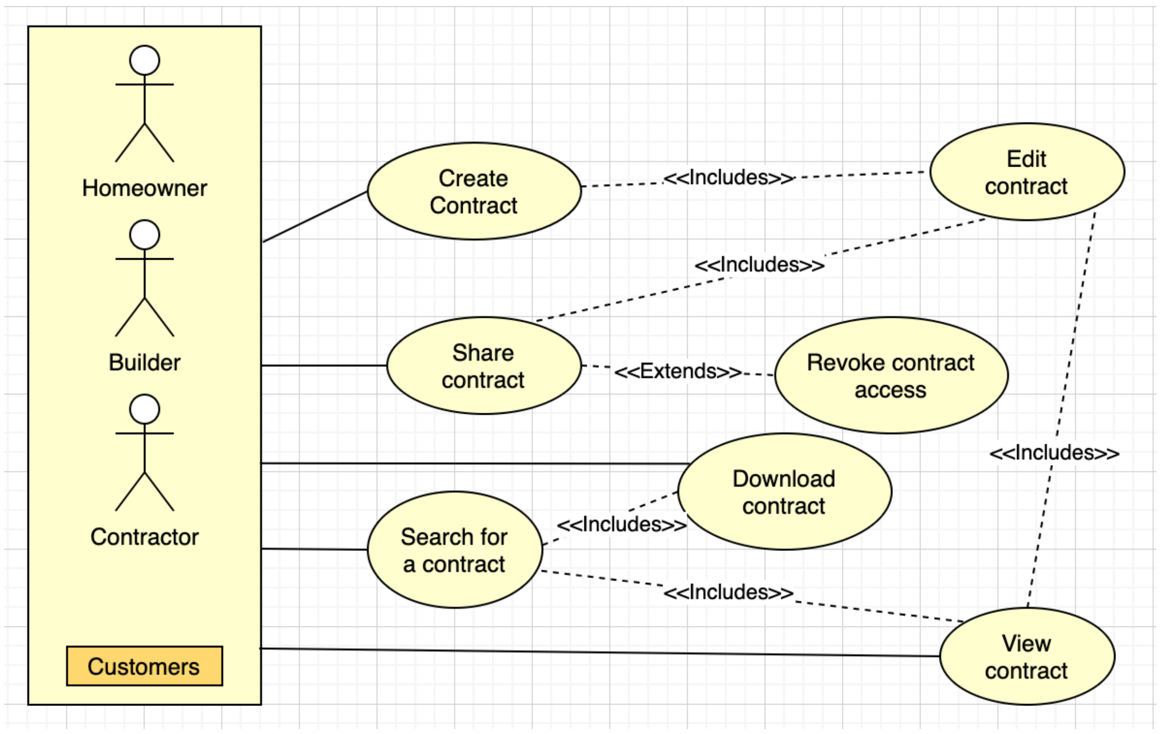


Figure 5: Use Case Diagram

Above is a use case diagram that shows how homeowners, builders, and contractors will be able to interact with our web app.

3.3.2 Functionality

Our design is a proof-of-concept web application that is accessible to BuilderTrend at any time. This app, combined with our extensive documentation, will provide BuilderTrend a clear path forward to fully integrate DocuSign with their entire platform. We intend to recommend implementations based on their business needs, as well as demonstrate how that implementation might work within our own web app.

This design should satisfy all of our functional requirements, since we can use the DocuSign API to clearly demo each use case that BuilderTrend is considering.

Our app can be implemented with little to no cost, and can satisfy non-fun ctional requirements such as speed and security. However, these don't necessarily translate to the non-functional requirements of BuilderTrend's final implementation.

3.3.3 Areas of Concern and Development

Our primary concern with our current design is security. Considering we are dealing with confidential contracts and information, we should prioritize security over accessibility and usability. Our current design utilizes the DocuSign API, which is known for its security, but we need to make sure all parts of our web app are secure.

Secondly, we are concerned about DocuSign's terms of service as it relates to our application. For our purposes, we are authorized to use the 'free' tier of DocuSign's API. However, there may be a limit on how many times we can call the API without having to pay. When BuilderTrend decides to implement DocuSign into their own API, they will have to pay for an enterprise license. This has been noted in our documentation.

3.4 Technology Considerations

When choosing our tech stack, we considered three main points:

1. What is our team familiar with?
2. What is BuilderTrend's most used tech stack?
3. What is most compatible with the DocuSign API?

We discovered that our team was most comfortable with JavaScript, Java, and C. Given that our product is a web application, the Frontend made sense to develop in JavaScript. React is a popular frontend framework for JavaScript that multiple team members had used in previous internships.

BuilderTrend's web products are built with a React JS frontend, and a C# backend. However, our team was much more familiar with withe Javascript than C#. For consistency, we chose to implement the backend in Node, a server-side JS framework.

DocuSign provides a comprehensive Node SDK for their eSignature API, as explained here. This will make for easier implementation as we tackle more complex requirements.

3.5 Design Analysis

While we are still early in the implementation stage of our project, our proposed design will work. We have conducted extensive research on the DocuSign API and its capabilities. As we get further into our implementation, we may tweak the frontend layout of our application. More hands-on testing will allow us to optimize the frontend pages for usability. In addition, we will continue to discuss our design with BuilderTrend to further tailor it to their specific needs.

3.6 Design Plan

The eSignature API Reference enumerates all the provided use cases, including how-to guides for our specific needs. For example, we will be utilizing the following methods from the official DocuSign Node SDK:

- `createEnvelope()` - Builder uploads a document
- `setTabValues()` - Builder identifies where to sign document
- `sendEnvelope()` - Builder sends documents to owners and subcontractors
- `createRecipientView()` - Owner views the document to be signed
- `getEnvelope()` - Any user views pending/signed documents on our application

- `getDocument()` - Any user views pending/signed documents on our application

These methods are documented here. Our implementation will call these methods from our Node.JS backend, which is an officially supported application type.

We will continue with our provided architectural diagram in 3.3.1. We will first focus on backend development, as the most important part of the project is to be able to interact with the DocuSign API in a meaningful way. Once we have a working backend, we will build out our local storage in a MySQL database, where we store any user information that is not kept by DocuSign. Then, we can implement a frontend that allows for users of all types to create, sign, and view documents in an easily accessible way.

4 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopting test strategy and instruments. In this overarching introduction, given an overview of the testing strategy. Emphasize any unique challenges to testing for your system/design.

4.1 Unit Testing

We will be using unit testing to prove that the code we are writing works properly before adding it to the code base. It will also ensure that no other parts of the code are broken as a result of any recent changes, and protects our codebase from regression. The primary tool that we will be using in our unit testing is mocking. Mocking will allow us to test individual functions in our code without having to worry about the test failing because an outside method isn't working properly.

4.2 Interface Testing

There are three interfaces within our design: The front-end, back-end, and the DocuSign API. The connection between the front-end and back-end will be automatically tested using unit tests and mocks, as well as functional tests. The same goes for the connection between the back-end and the DocuSign API.

4.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

The standard need for integration testing does not directly apply to our project. Generally, integration testing is done by adding a new component into the greater system and using a service that is capable of imitating the actions an end user would take. Thus verifying that the new module is functional as a part of the whole. Our project is to provide a proof of concept for BuilderTrend to verify the viability of DocuSign. Thus they will be the ones integrating it into their systems.

Integration testing is not necessary as we are not “integrating” the project into something else. We will of course be performing system testing which is similar to integration testing, but limited to the project itself. Information on that may be found in the following section.

4.4 System Testing

To make sure that a complete system works, we will need a set of tests (unit, interface, and integration tests) that show complete functionality over the use cases described in the original plan. Main use cases that were listed in the requirements documents are going to be tested with unit testing. This is to verify that the main use cases are working correctly with use cases that are independent of each other. For our project the main use cases would be the following:

Have a way that a builder + contractor can electronically sign a document

Allow multiple contractor signatures via the Docusign API

Have a way that a homeowner + builder can electronically sign a document

Thorough documentation about DocuSign’s API and the use case solutions

With interface testing, we verify that use cases that have common functionality with each are working. In this area of testing, we may also identify places in the code where duplicate functionality exists and combine these into one piece of code. While we dont expect that to happen, it could happen because of the concurrent development that we have. In a more specific example, we will expect to see the integration of a front-end and back-end interface with our software. This is described more in the interface section of this document above.

While our project may not have integration testing, if we eventually get to the Buildertrend source code portion of the project, we may expect the need for integration testing with systems that Buildertrend already has to validate a working system.

4.5 Regression Testing

We will ensure that any new additions do not break functionality by breaking up our code into separate units. Each unit can then be modified and add the capability to use other units. By making sure that each unit talks and communicates with each other separately, we can then integrate new features without worrying about breaking old interfaces. We need to ensure that the requirements BuilderTrend wants are kept working while adding new features if they request them. To add, all of this is what will happen if BuilderTrend requests we integrate anything with their code. This is because they want us to deliver a working model to show itIs feasible. We will use mocking as stated above as our tools of testing.

4.6 Acceptance Testing

All requirements will be fully visible and usable in our minimum viable product - the web application. This will demonstrate the functionality that BuilderTrend requested, as well as the documentation to support the decisions we made and our recommendations for the future. We will present this to BuilderTrend, and ask that they test all of our functionality to see if it fits their requirements. We expect this to be an ongoing process next semester, so acceptance testing will consist of real usage and feedback from our client.

4.7 Security Testing

Some of the main threats to our React application include Cross-Site Scripting (XSS), SQL injections, and XML External Entity Attacks (XXE). We plan to manually execute these attacks on our product to ensure they are not possible. Security vulnerabilities related to the DocuSign API is slightly out of the scope of our project, considering DocuSign is a well known and trusted company.

4.8 Results

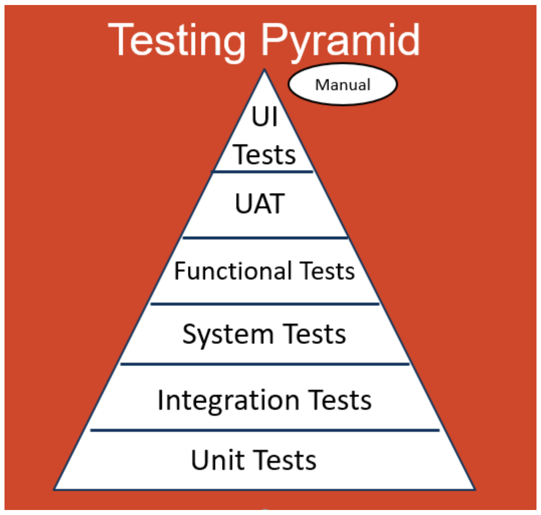
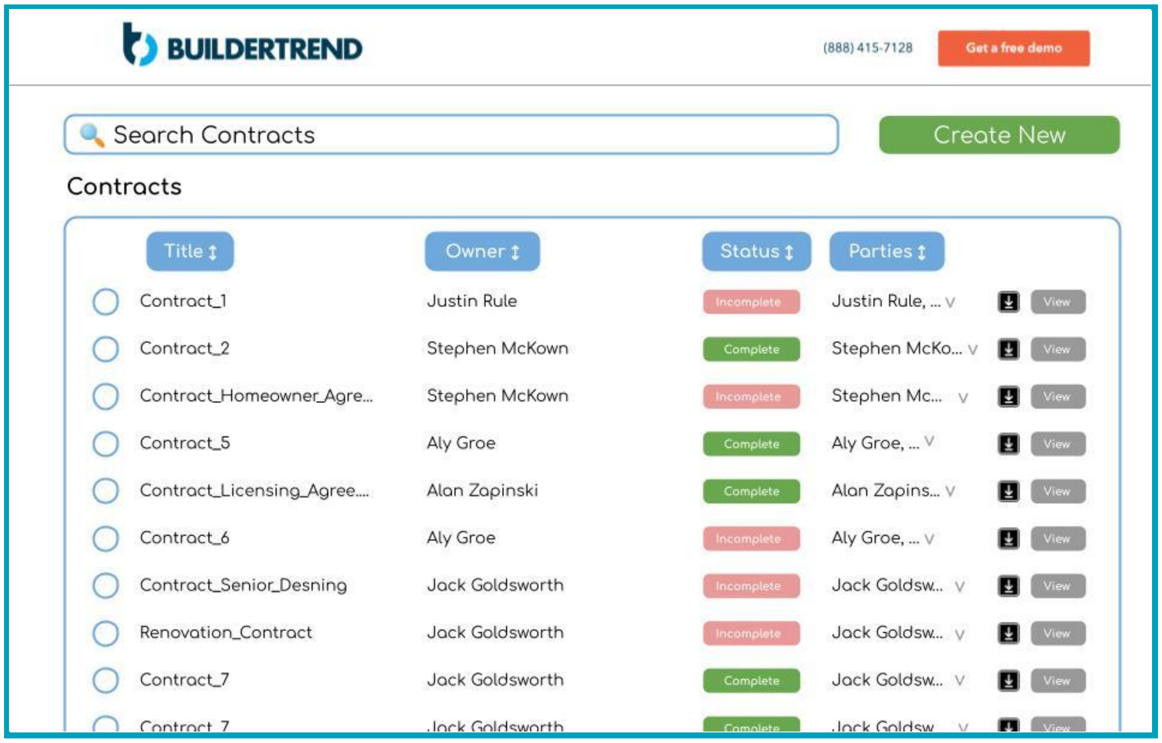


Figure 6: Testing Pyramid

Testing ensures that our project satisfies all the requirements specified by our client, and does so efficiently and as expected. By following a pattern similar to the testing pyramid above, we can easily verify all the use cases that BuilderTrend has mentioned. Also, we can more easily identify issues based on which layer of tests they affect. In the pyramid, the UI Tests layer maps to our acceptance tests - where real users (developers and clients) can test our application to ensure that it satisfies all requirements. This will be updated when we are further into the development phase.

5 Implementation

There has been little implementation of the project plan. We did create the following mockup of what a user interface might look like after project completion to help unfamiliar parties grasp what the end goal is.



Contracts

	Title ↕	Owner ↕	Status ↕	Parties ↕		
○	Contract_1	Justin Rule	Incomplete	Justin Rule, ... v	⬇	View
○	Contract_2	Stephen McKown	Complete	Stephen McKo... v	⬇	View
○	Contract_Homeowner_Agre...	Stephen McKown	Incomplete	Stephen Mc... v	⬇	View
○	Contract_5	Aly Groe	Complete	Aly Groe, ... v	⬇	View
○	Contract_Licensing_Agree....	Alan Zapinski	Complete	Alan Zapins... v	⬇	View
○	Contract_6	Aly Groe	Incomplete	Aly Groe, ... v	⬇	View
○	Contract_Senior_Desning	Jack Goldsworth	Incomplete	Jack Goldsw... v	⬇	View
○	Renovation_Contract	Jack Goldsworth	Incomplete	Jack Goldsw... v	⬇	View
○	Contract_7	Jack Goldsworth	Complete	Jack Goldsw... v	⬇	View
○	Contract_7	Jack Goldsworth	Complete	Jack Goldsw... v	⬇	View

Figure 7: Proposed UI Mockup

For this project, implementation is entangled into design as any implementation decisions needed to be considered as part of the design. You can find those details in the design section.

6 Professionalism

This discussion is with respect to the paper titled " Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

6.1 Areas of Responsibility

We chose the SE code of ethics.

Work Competence: This is covered in Principle 2 of the SE code of ethics, as it promotes honesty about one's limitations and work they can do. Principle 8 (Self) also promotes self improvement and knowledge about an engineer's areas of expertise.

Financial Responsibility: Principle 4 (Judgement) relates to financial responsibility as it talks about professional objectivity, deceptive financial practices, and proper disclosure of billing. Both the SE code and NPSE cite trust and honesty in financial agreements.

Communication Honesty: Principle 7 (Colleagues) and Principle 5 (Management) speak on appropriate interactions between management and developers and in between colleges. Highlighting the importance of ethical interactions and management of software products. Included in this is honesty between levels of management and within the project itself.

Health, Safety, Well-Being: Principle 7 (Colleagues) and Principle 2 (Client and employer) discuss the importance of respecting your colleagues and acting in the best interest of the company and the clientes. These don't relate very strongly to Health and Safety as software engineering is not particularly dangerous, but morals must be kept in mind.

Property Ownership: Principle 2 defines a good relationship between engineers, clients, and employers. It requires that confidential information be kept private and that engineers should not pursue efforts that compromise their employer's interests. It also discusses the proper use of software - an engineer should not use illegal software or software without authorization.

Sustainability: The SE code doesn't specify much about environmental sustainability, but Principle 3 (Product) mentions creating high quality, sustainable software.

Social Responsibility: Principle 1 (Public) describes a fair approach to disclosing work to the public and making socially responsible decisions within software. It also asks that engineers take full responsibility for their work. This is important to the interests of the business as well as the users. The SE code of ethics goes into more detail about creating a good, moral user experience while the NPSE version wants to "enhance the reputation of the profession".

6.2 Project Specific Professional Responsibility Areas

Work Competence. Yes, every member of the team has completed at least one internship with a professional software company. We have a high level of experience with the required skills and are capable of carrying out a plan to completion in a way that will lead to success. Every team member has already proven valuable to our team and has shown what they are capable of thus far.

Financial Responsibility. Not directly, by BuilderTrend giving us specific deliverables, the burden of financial responsibility does not rest with us as we did not choose what our end product would be. Additionally, no team members are being paid, and other than bi-weekly meetings with the BuilderTrend liaison, our project will have no costs for BuilderTrend.

Communication Honesty. Yes, this is likely our most important **professional responsibility area**. The success of this project is dependent on us taking full advantage of the resources provided such as our Faculty Advisor and our BuilderTrend liaison. Without honesty and communication, failure is almost inevitable. The current performance is not indicative of our long term success in this area, but the high level of **work competence** on the team suggests that we will honestly communicate with our connections.

Health, Safety, Well-Being. Somewhat. This project will have a neutral to positive impact on stakeholders. It is highly unlikely that events that occur will have a negative effect on stakeholders. Public safety will inevitably be improved as the project directly reduces the personal contact between people and increases communication and understanding between parties. Relative to the

professional responsibilities of the team, any successful completion of the project will result in a positive impact on the public and the company. Thus, there are limited specific actions needed to be carried out by the team other than completing the project.

Property Ownership. Marginally, BuilderTrend and Docusign have provided resources for us to use, but none of the resources are consumable or damageable. The project itself will need to protect the information of BuilderTrend and their customers. This is a concern we have held in mind throughout the entire design process. During project completion, we will not have access to confidential data. This significantly reduces the risk of a data breach and thus reduces the severity of compromising actions a team member could accidentally take. Thus the only risk we face in this **professional responsibility area** is the project itself and any related documentation.

Sustainability. No, this project is entirely electronic, and will not be irresponsibly electrically demanding. In fact, completion of this project will reduce paper use and the need for physical delivery of documents.

Social Responsibility. Yes, completion of this project will improve the lives of those who use it, and will have no impact on those who don't. The project will need to ensure the safety of confidential information. This is something that we have been aware of throughout the design process.

6.3 Most Applicable Professional Responsibility Area

One area that is important to both our project and our team is work competence. This means a lot to our project because we want to keep BuilderTrend's customer's information secure as they interact with each other using the new DocuSign API's. We will be delivering work of high quality and integrity by making sure to securely store and manage customer's information within our deliverable by encrypting their data.

Although we might not deal with sensitive information directly, we are providing a complete blueprint and proof of concept for Buildertrend's DocuSign integration. Thus, it is necessary to provide a safe and secure solution, while also documenting possible risks related to security. Mainly, this would involve securing customer's information and contractor's information while sending contracts to and from each other. Then, Buildertrend can confidently implement our solution into their products knowing that we have accounted for data security. We also want to make sure that DocuSign's API's are also secure and that we interchange information with them securely as well.

7 Closing Material

7.1 Discussion

We have designed our solution to meet the requirements that our client brought to us. We incorporated the fact that there can be multiple signers into our project. We have also included a menu where you can view the status of the signatures requested.

7.2 Conclusion

We started by researching the DocuSign API to confirm that the DocuSign API is what we (and our client) wanted to be using. Then based on what the API provided, we decided on our foundational programming languages and frameworks. After this, we continued to look into the DocuSign API to discover it's limitations and unknown areas that could benefit us.

To reiterate, our goal is to design a prototype application that shows how DocuSign can be used to implement contract signatures through a third party application. We will then thoroughly document this implementation and give steps on how we implemented our solution. This will go to BuilderTrend, where they can then implement our solution within their codebase.

The best plan of action to achieve these goals would be to continue on the path that we have been going down for this semester. We have researched the DocuSign API and we have decided on the surrounding technologies to produce our prototype. We have even created a demo application as a proof of concept. Next semester, we will use this learned knowledge to product our prototype using the foundation we set this semester.

7.4 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.

7.4.1 Team Contract

Team Members:

1) Jack Goldsworth________________	2) Carson Meyer_____________________
3) Ale Groe________________________	4) Justin Rule____________________
5) Conley McKown________________	6) Joe Slater______________________
7) Alan Zapinski__________________	8) _______________________________

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

Virtual - Sunday Afternoons

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-

mail, phone, app, face-to-face):

IM Apps - GroupMe/ Discord

3. Decision-making policy (e.g., consensus, majority vote): **Consensus**

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be

shared/archived):

Rotate every week, they will take notes throughout the meeting which will be shared to google drive for everyone to access.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

We expect all meetings to be attended unless specified otherwise. If you are late more than 3 times, we will meet as a team and figure out how to address the situation.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

We will strive to complete all work assigned to ourselves before the deadline. If work is finished after the deadline, we will meet as a team and discuss how to better help each other out in the future.

3. Expected level of communication with other team members:

All team members are expected to communicate any upcoming absences so meetings can be run based on who will be attending.
Any important decisions will be made in a meeting with everyone's attendance

4. Expected level of commitment to team decisions and tasks:

All team members should commit fully to their tasks and be able to complete them in a timely manner.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction,

individual component design, testing, etc.):

Jack - General testing
Justin - Individual component testing
Ale - Project manager
Conley - Technical requirements
Carson - Professor / TA communications
Joe - Client interaction
Alan - Team relationship mediator

2. Strategies for supporting and guiding the work of all team members:

If anyone has questions, they should feel open to ask any team member for help. Other team members are highly encouraged to respond within one day.

3. Strategies for recognizing the contributions of all team members:

Team members will talk about what they worked on since the last meeting so proper recognition can be granted to different features of the project

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the

team.

Alan: Python, CI-CD process, AngularTS, Git, Rest APIs
Justin: Frontend development, Java development, Git, Consumer facing development, SQL, REST APIs.
Ale:Java, ReactJs, Frontend development, Git, SQL, C, Rest APIs.
Joe: Java, C/C++, JS, React, Ionic, Git, Android Studio, Frontend development
Jack: Java, C/C++, Kotlin, Python, JS/TS, React, Git, AWS, Git
Conley: Java, C/C++, Python, Git, Android Studio
Carson: Java, C/C++, Python, Git, SQL, Microsoft Dynamics AX 2012, Backend development

2. Strategies for encouraging and support contributions and ideas from all team members:

All team members will help brainstorm and come up with ideas for how to complete our project. We will make sure to ask our peers if they have any questions or concerns. Anyone is free to suggest an idea and have a chance to explain it as well.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will

a team member inform the team that the team environment is obstructing their

opportunity or ability to contribute?)

Professionally, if someone has a problem with the team they will be forward with the group and express their opinions and concerns. From there we will discuss as a group how we can resolve the conflict and decide if roles need to change or something similar.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Complete all assigned tasks in a timely manner, while including and hearing input from each team member.

2. Strategies for planning and assigning individual and team work:

Meet weekly and keep each other informed of your progress.

3. Strategies for keeping on task:

Weekly standup to make sure our team is on task and completing their work.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

We won't punish the first offense of any individual obligation, but upon repeated infractions, we will have a discussion with the team member about this contract.

2. What will your team do if the infractions continue?

If the issue does not resolve, we will contact our professor or TA to discuss next steps. These may include but are not limited to: termination.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the

consequences as stated in this contract.

1) Jack Goldsworth DATE __12-04-2021________

2) Carson Meyer DATE __12-04-2021________

3) Ale Groe ____ DATE __12-04-2021________

4)Justin Rule_____________________________________ DATE ____12-04-2021______________

5) Alan Zapinski_______________________________________ DATE 12-04-2021__________

6) Joe Slater____________________________________ DATE 12-04-2021_____________

7) Conley McKown_______________________________________ DATE 12-04-2021_________

8) ____ DATE __________________