Final Presentation Assurance Recipes Application

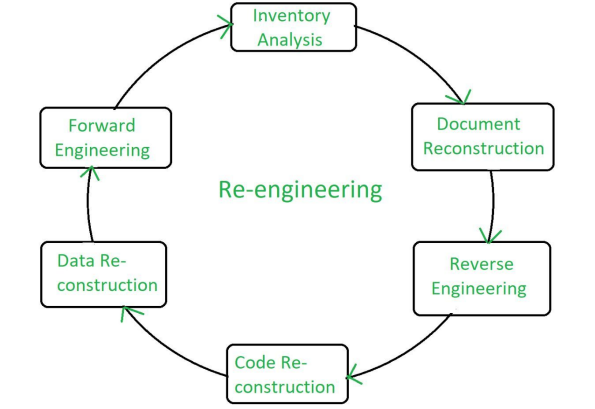
Group Number: sdmay20-26
Team website: https://sdmay20-26.sd.ece.iastate.edu/
Client and Advisor: Myra Cohen
By: Matthew Smith, Kevan Patel, Qiwei Li, Shiwei Wang, and Garrett Harkness

Acknowledgements

- Myra Cohen - Our client and advisor
- Justin Firestone - Myra’s grad student and helper of creating accurate Assurance Recipes Application
- Mark Hernandez - The original programmer of the Assurance Recipes prototype application

Motivation

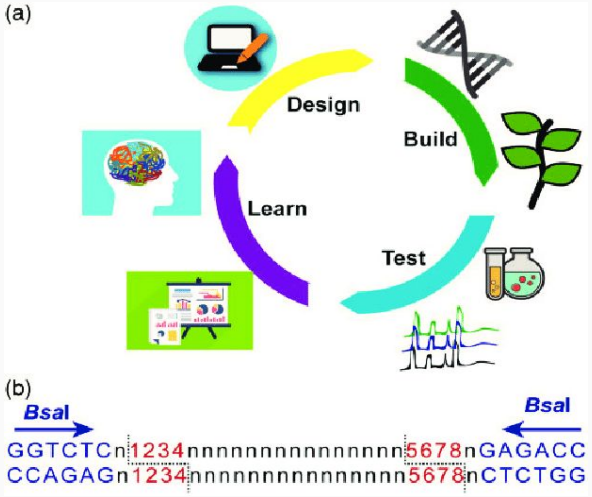
- We are recreating an app previously made by a student named Mark Hernandez.
- The application will be used by synthetic biology students to participate in a large competition called iGEM.
- It will also be used by professional synthetic biologists.



What is Synthetic Biology?

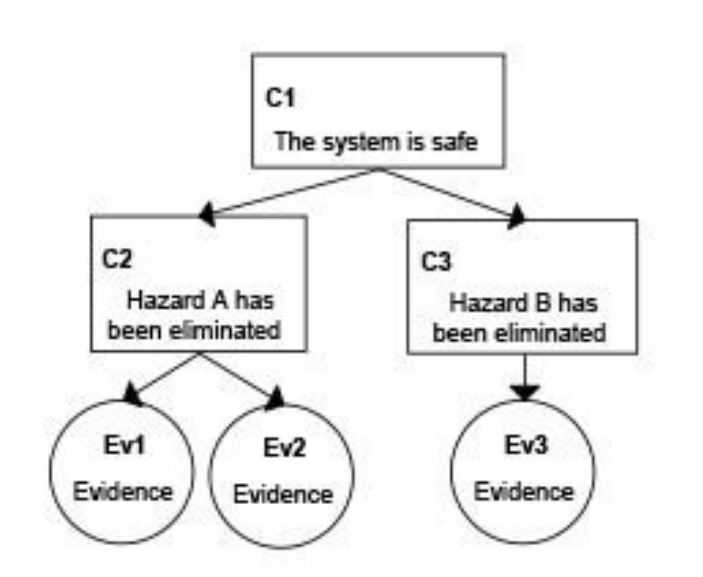
- Synthetic biology involves redesigning organisms for useful purposes by engineering them to have new abilities.
- For example E. Coli.
- Many aspects of synthetic biology is also similar to how software is created.

E. Coli (Above Figure)



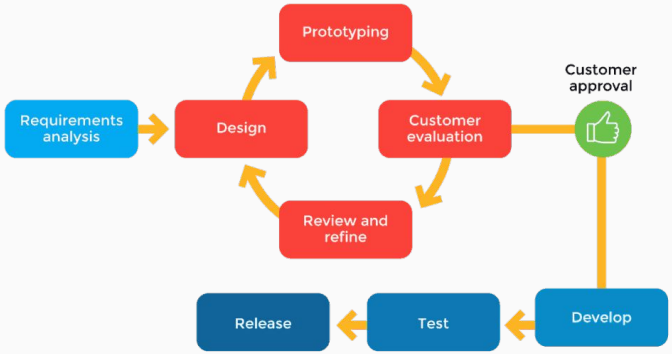
The Assurance Recipes Application

Using a notation called Goal Structuring Notation (GSN) which was chosen by our client, Assurance Recipes and Cases are a concept in synthetic biology to allow people who use genetically modified organisms to feel confident that what they are testing and using is safe. Our application will help allow students and researchers to create and outline diagrams in a fully implemented design.

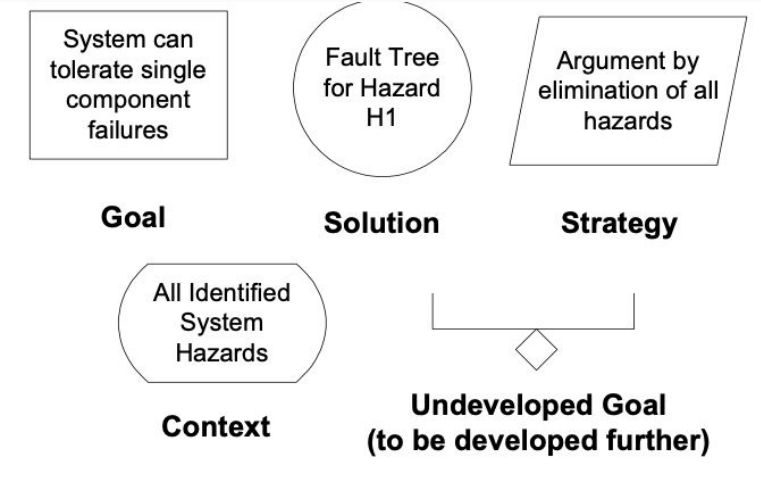


Objective and Problem Statement

- The original website was coded by Mark Hernandez - Meant to be temporary.
- Uses an unsafe database that isn't owned by our client.
- Was used and meant to be a prototype used on for the iGEM competition.
- Hosted on website that needs to be moved.
- We want to recreate and redo the webapp so that it can be released to the community.



Goal Structuring Notation (GSN) and Assurance Cases

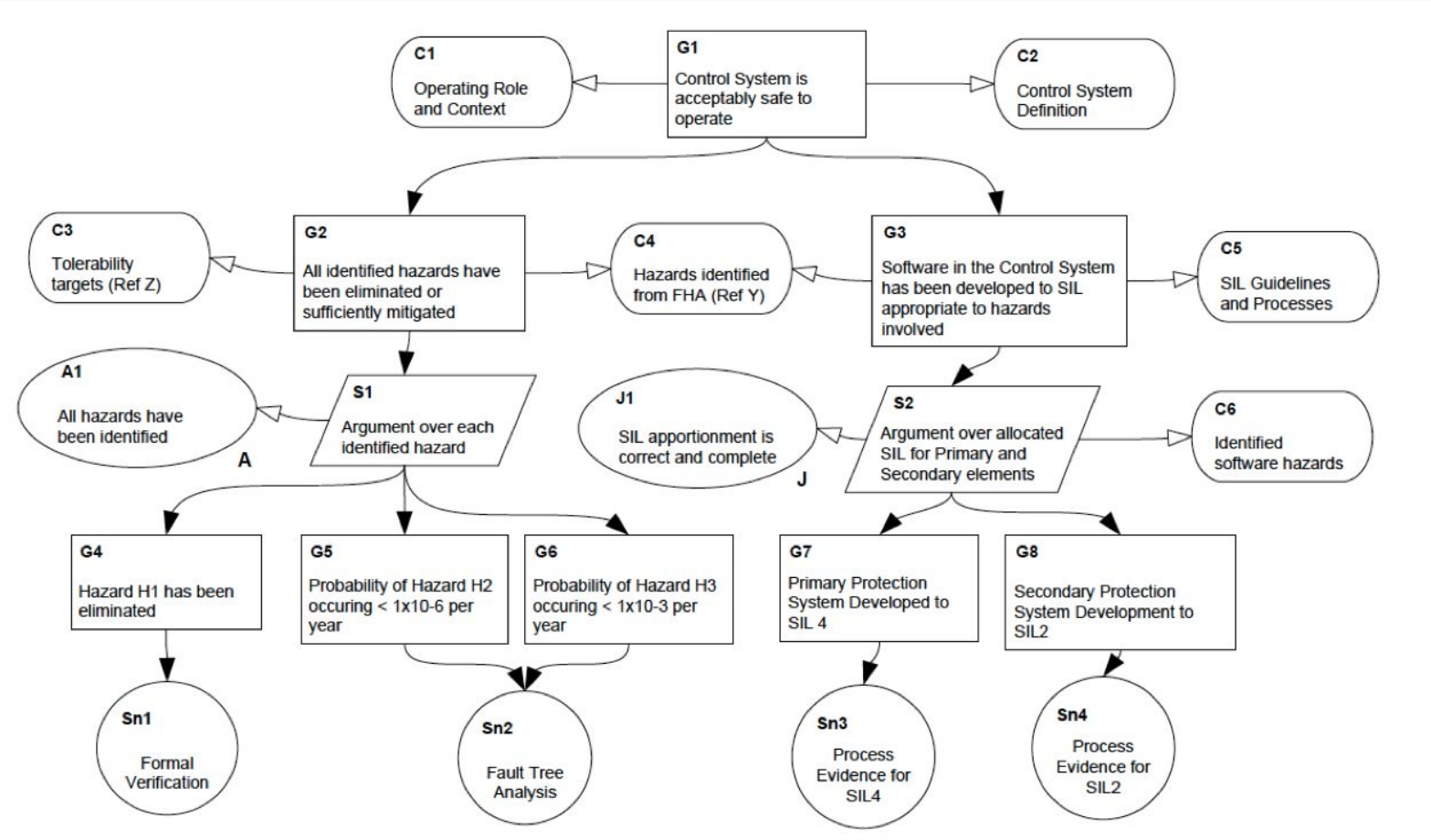


- Set of arguments and corresponding bodies that provide evidence to these arguments.
- Applicable across many different industries.
- Complex systems lead to complex assurance cases.
- Drawbacks:
  - Can become too complicated to users who lack expertise.
  - Can have too many degrees of freedom.

The Goal Structuring Notation – A Safety Argument Notation by Tim Kelly and Rob Weaver of the University of York

International Genetically Engineered Machine (iGEM) Competition

- All projects are open source.
- Safety of projects must be explicitly discussed.
- Gold Medals require teams to accomplish Integrated Human Practices
    - Asks teams to “consider whether their projects are safe, responsible, and good for the world.”



Assurance Case Example from https://www.cs.york.ac.uk/assure2013/ASSURE2013Slides/0ASSURE_SC_Intro_Final.pptx.pdf

Assurance Recipes

- Abstraction of assurance cases.
- Template-Like Model
- Provide user with structure/pattern for assurance case, user selects ingredients.
- Customized for domain, then parameterized for easy user instantiation.
- Containment Recipe
- Safety Mechanism Recipe

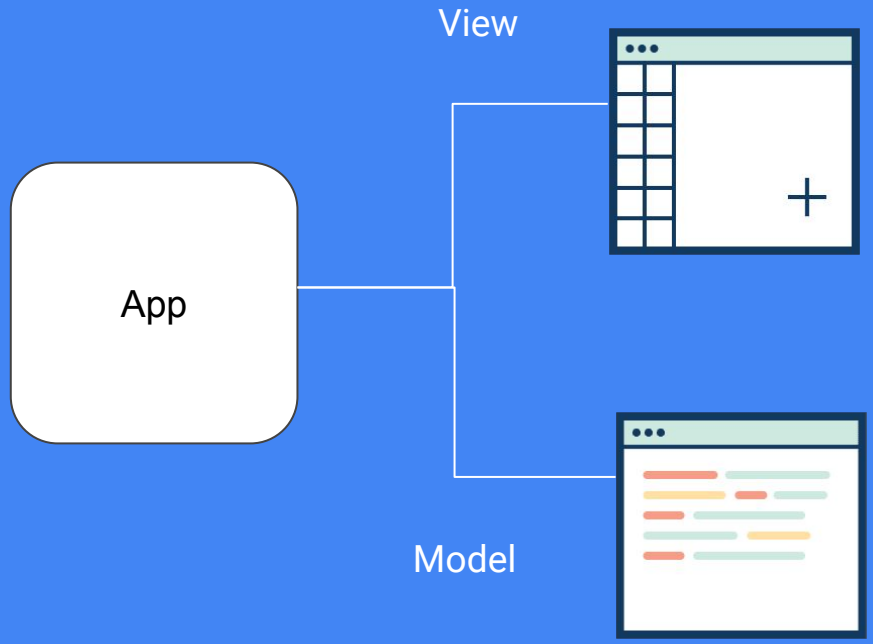
Past Literature

- The Assurance Recipe: Facilitating Assurance Patterns
  - Myra Cohen, Justin Firestone
- The Goal Structuring Notation – A Safety Argument Notation
  - Tim Kelly, Rob Weaver

React

A JavaScript library for building user interfaces

React can apply MVP model for UI development by using React Context API



Electron

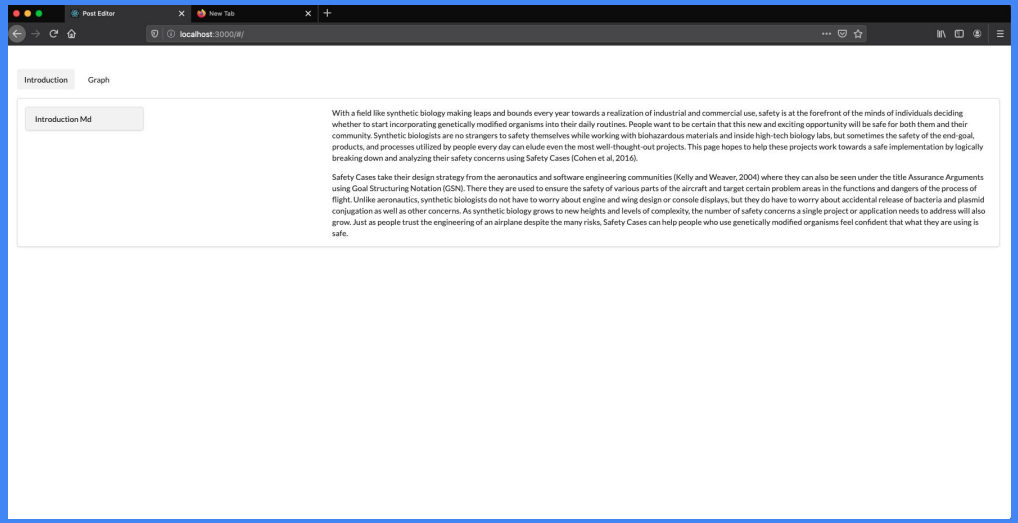
The Electron framework lets you write cross-platform desktop applications using JavaScript, HTML and CSS. It is based on Node.js and Chromium and is used by the Atom editor and many other apps.

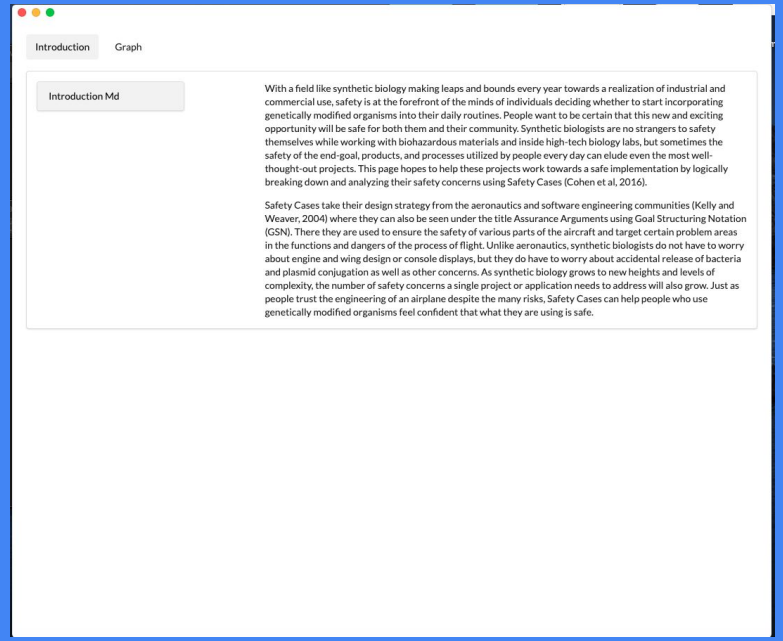
- Context menus
- Tabbar menus
- Drag and drop
- Local notification
- Directly modify local files

Our client asked us

- Both Web App and local App
- One single code base
- Works on every platform

Electron is the best option!





Prototype Browser version and Electron version

Using Electron builder to generate app

All platforms: 7z, zip, tar.xz, tar.lz, tar.gz, tar.bz2, dir (unpacked directory).

- macOS: dmg, pkg, mas, mas-dev.
- Linux: AppImage, snap, debian package (deb), rpm, freebsd, pacman, p5p, apk.
- Windows: nsis (Installer), nsis-web (Web installer), portable (portable app without installation), AppX (Windows Store), Squirrel.Windows.

Others

- Code Gen (Documentation)
- Knova JS
- Selenium (Testing)
- React Testing Library (Testing)
- JSON Schema Generator (Form)

Docker

Debug your app, not your environment

Securely build, share and run any application, anywhere


```
1  FROM node:12.13.0-stretch
2  WORKDIR /home/projects
3  COPY package.json package.json
4  RUN yarn
```

Dockerfile we used for this project

Functional requirements

- Creation and editing of safety case diagrams.
- Creating safety cases based on a template.
- Free text editing on the diagrams.
- Importing and exporting of safety cases on local machine.
- Opening an existing safety case.

Non-functional Requirements

Performance: Running the application in a short response time.

Scalability: Users being able to add assurance cases.

Security: Saving and creating in a secure environment. No sensitive information being leaked.

Availability: An app that is available for those who need to use it.

Reliability: Safe stable software

Usability: Easy to use and understandable interface for users not well versed in a technical background.

Test Plan

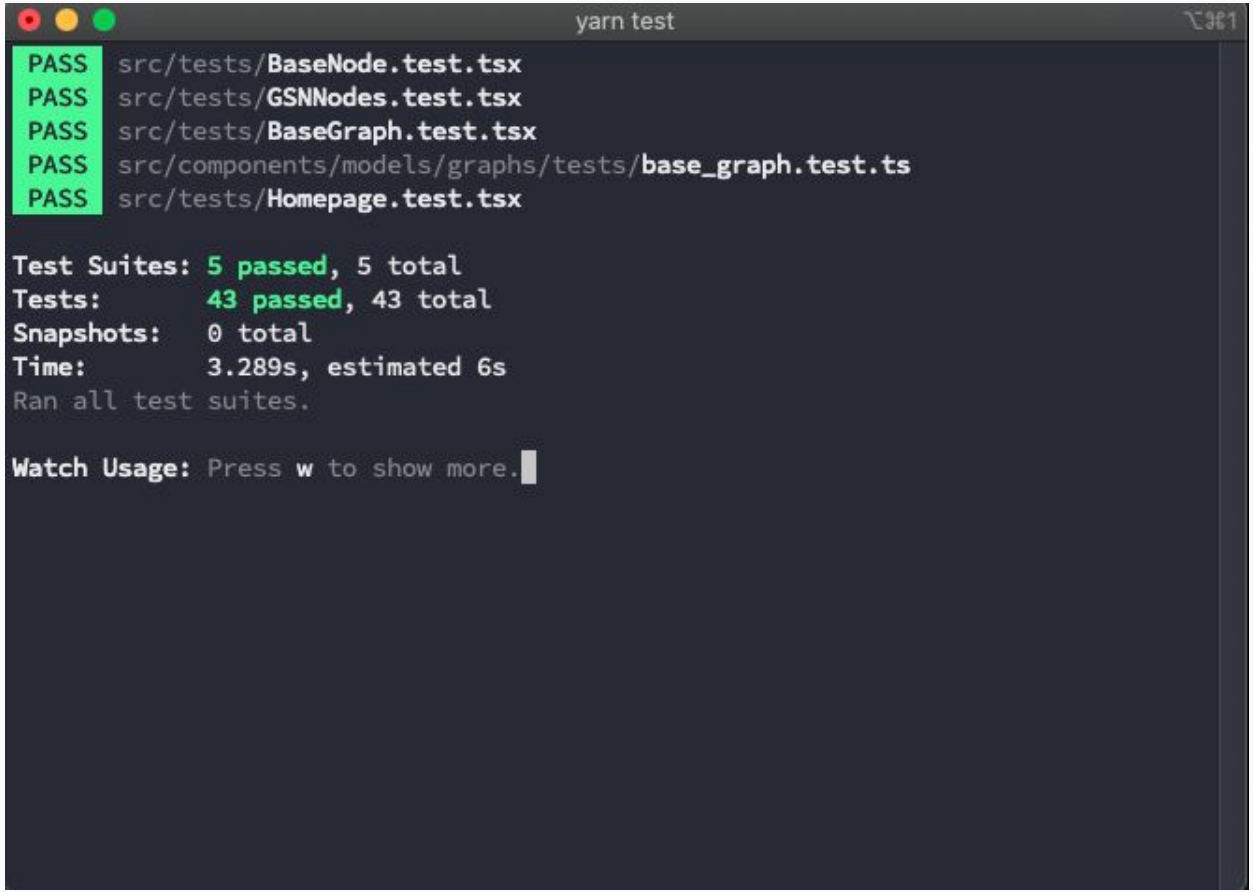
- Interface Specifications
- Functional Testing
  - Testing models
  - Testing UI
- Non-Functional Testing
  - Performance
  - Security
  - Usability
  - Compatibility

```
describe("Test utils", () => {
    const rsp: Result<Post> = { count: 1, results: [{ title: "Some title",
content: "1234" }] };
    (axios.get as jest.Mock).mockResolvedValue({ data: rsp })
    test("Test search function", async () => {
        let result = await searchPost("Hello")
        expect(result.count).toBe(1)
        expect(result.results).toBe(rsp.results)
    })


})
```

Example of model testing

Testing Interface



Risks and Costs

- Store user’s data
- No costs for any hardware and other sources
- Working remotely and Pandemic

Project Milestones & Schedule & Planning

2019

Basic Template Setup

Choice libraries. Setup project template

2019

Implement basic application

Implemented basic functions. Added unit tests

2020

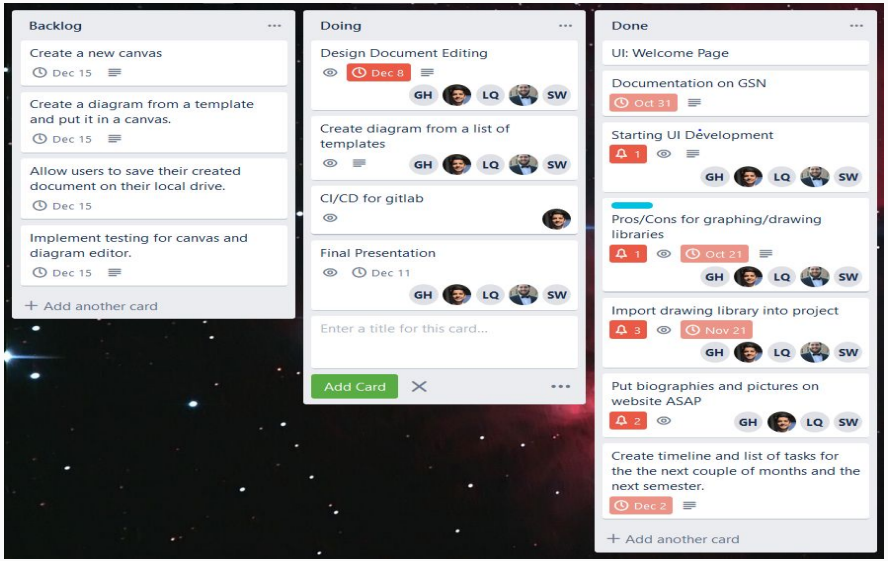
Basic Function

Finished the basic functions' implementation. Add more tests.

2020

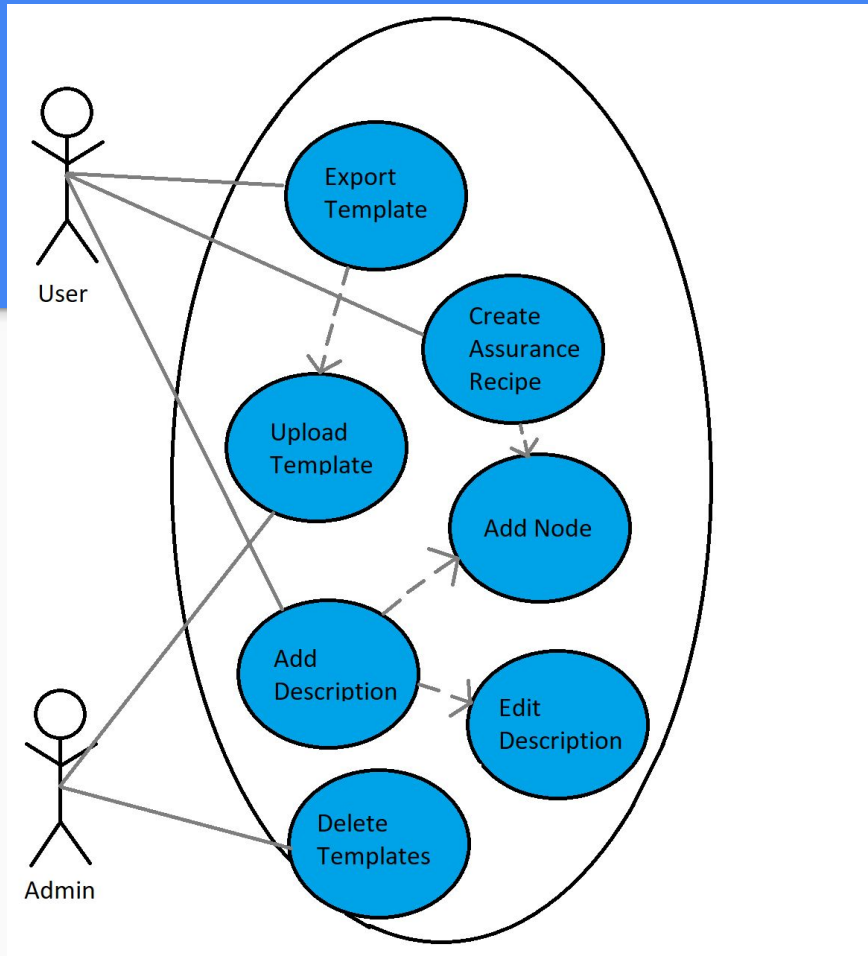
Final Product

Cleaning up final project for a presentable final application

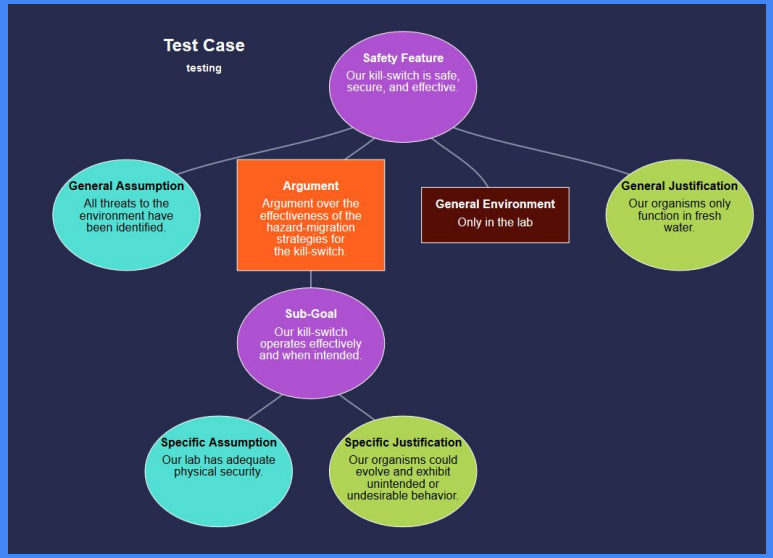


Functional Decomposition

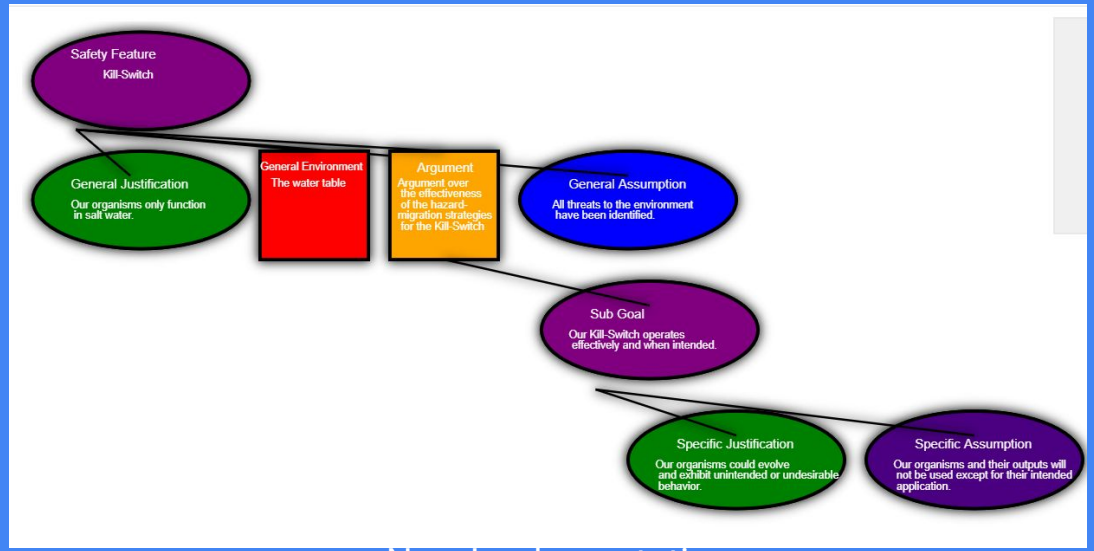
User:
-Export Template
-Create Graph
-Add Node
-Delete Node
-Add description
-Delete description
Admin:
-Delete template
-Upload template



Prototype Implementations

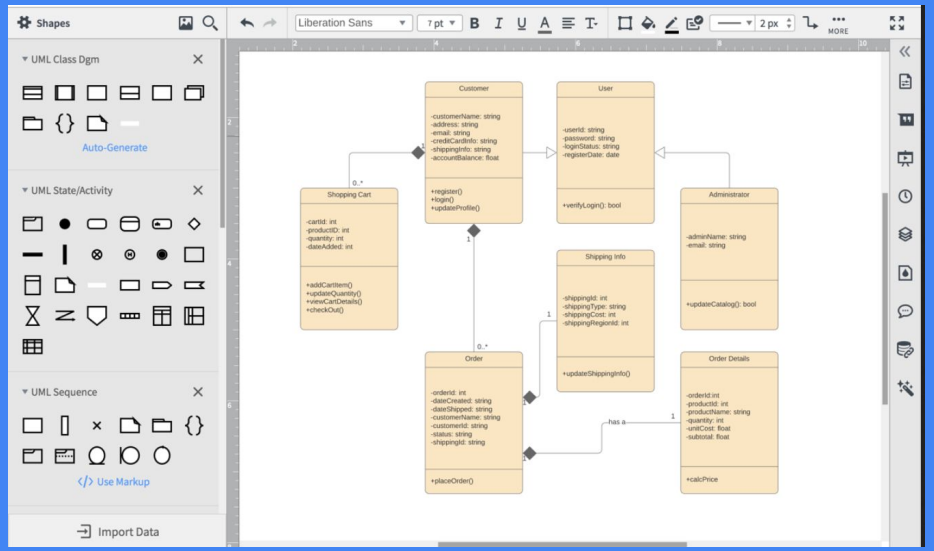


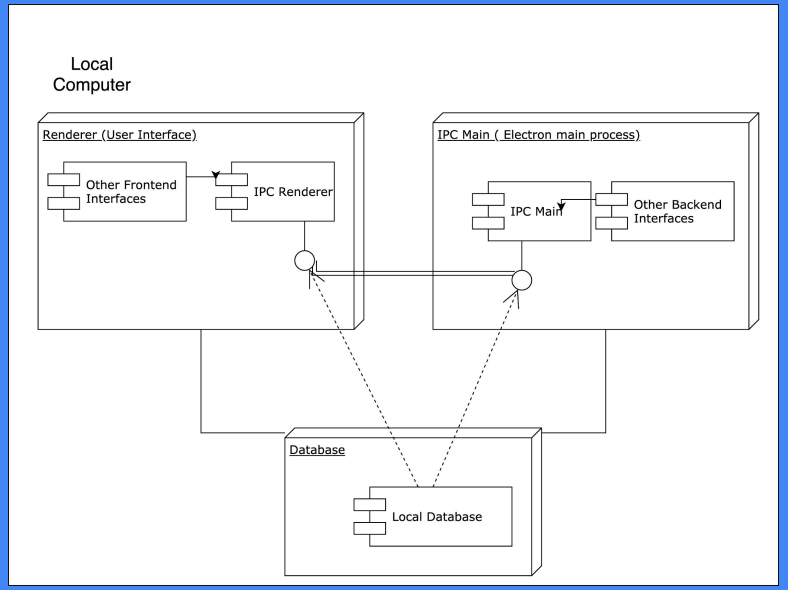
Old Implementation



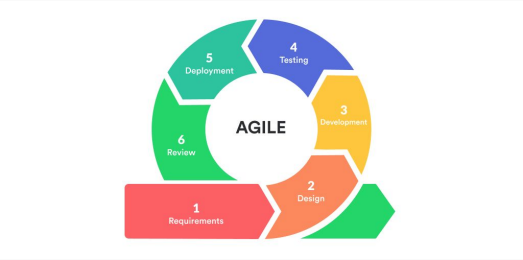
New Implementation

Detailed Design



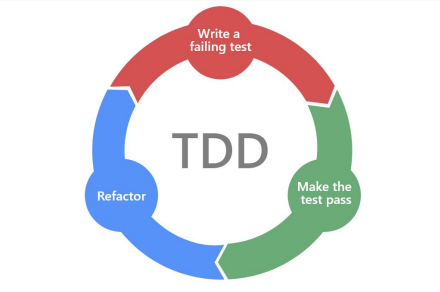


Engineering Standards and Design Practices



DISCORD

GitHub



Webex Meetings

slack

Task responsibility

For Group working:

- Weekly status report
- Lightning talk
- Design document
- Weekly meeting with client
- Weekly group meeting

Matt: Meeting Facilitator

Garrett: Scribe/Test Engineer

Kevan: Report Manager

Qiwei: Chief Engineer

Shiwei: Test Engineer

Thank you!