Introduction

XRP ofers a flexible and adaptable platform that allows students to learn about robotics, programming, and engineering concepts at their own pace and it also enables STEM education accessible to a wider range of students, regardless of their background or location.

In addition, the goals are to inspire and empower young people to pursue careers in STEM fields through hands-on, engaging robotics experiences. Encourage students to work together to solve challenges and develop innovative solutions and an cooperative environment where students can fosther grow individually and professionally.

The student uses this Web application platform along with the XRP Robot to perform their programming, troubleshooting and experimentation tasks.

Scope

Sparkfun manages the development of the XRP controller board in collaboration with WPI and community volunteers. The XRP Web application is a collaboration between WPI and community volunteers. This document describes the software architecture of the XRP software platform including both the Web application and the firmware release to the XRP hardware platform.

Software Architecture

Quality Attributes

The software architecture intends to focus on the following architecture principles.

- Maintainability
- Testability
- Extendability
- Adaptability
- Developability

Constraints

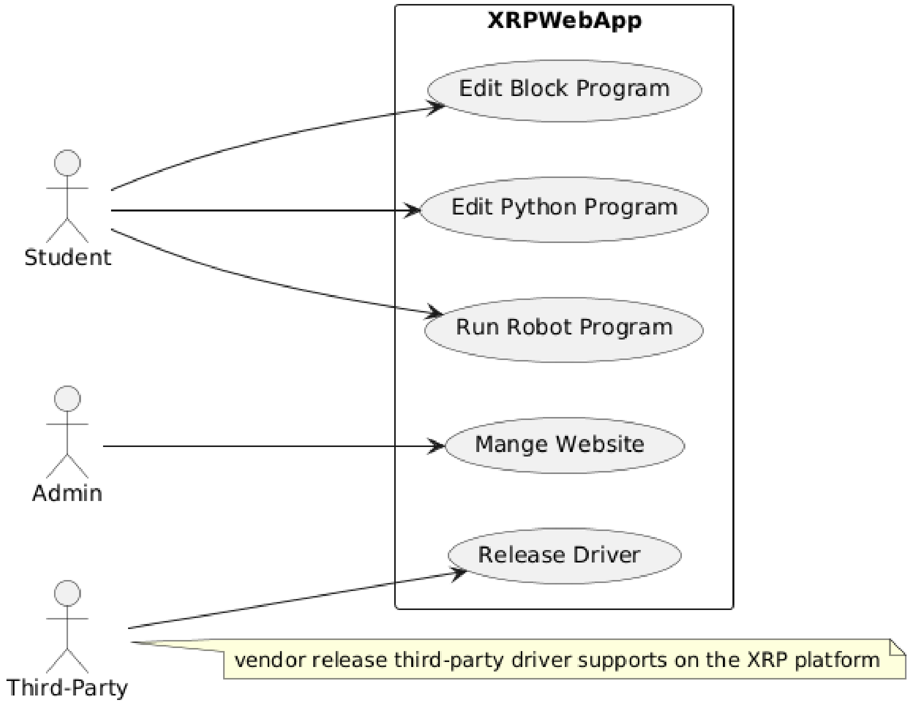
The XRP software architecture shall embrace the software component design patterns and declarative user interface paradigm to enhance code readability and maintainability.

The XRP platform is an Open Source and it is not a for profit software product. The development environment leverages mostly Open Source software development tools such as Visual Studio Code and Micro Python and Open Source software libraries.

Vite and React can satisfy both the software component design pattern and declarative user interface design paradigm.

Use Cases

The diagram below depicts the use cases for the XRP Web Development Application.



Technology

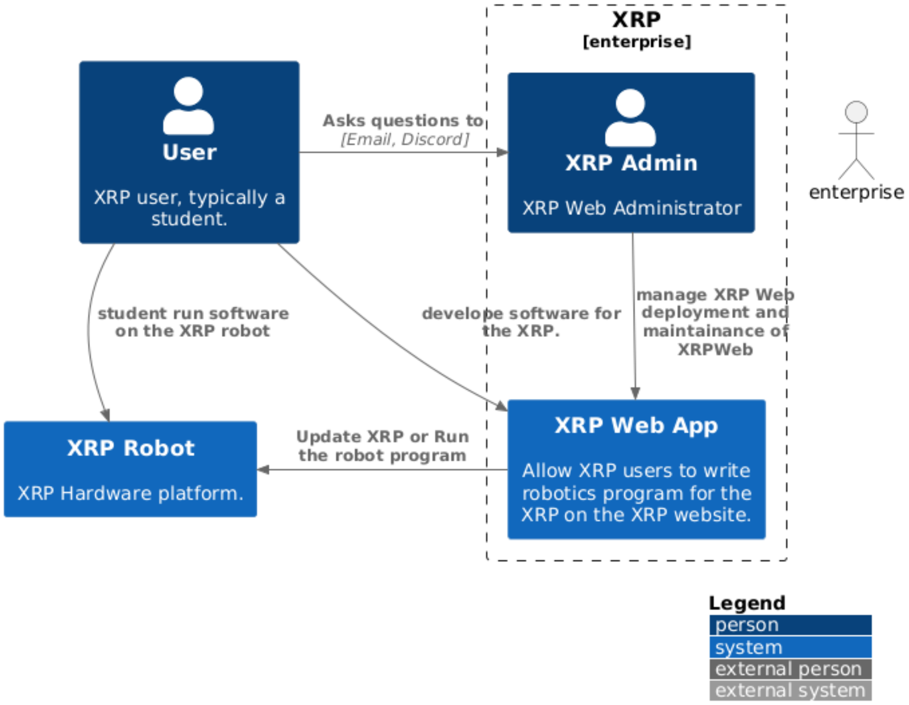
This software architecture specifies the following technology stack for development.

- Vite Development framework and React User Interface framework
- Typescript programming language
- HTML/CSS (Tailwindcss)
- Web browser serial protocol

- JSON
- Bluetooth (BLE) protocol
- HTTPS protocol

Context

Experiemential Robotics Platform (XRP) is a learning platform consists of the XRP hardware that is based on Sparkfun's controller board and XRP software development environment for writing software for the XRP.

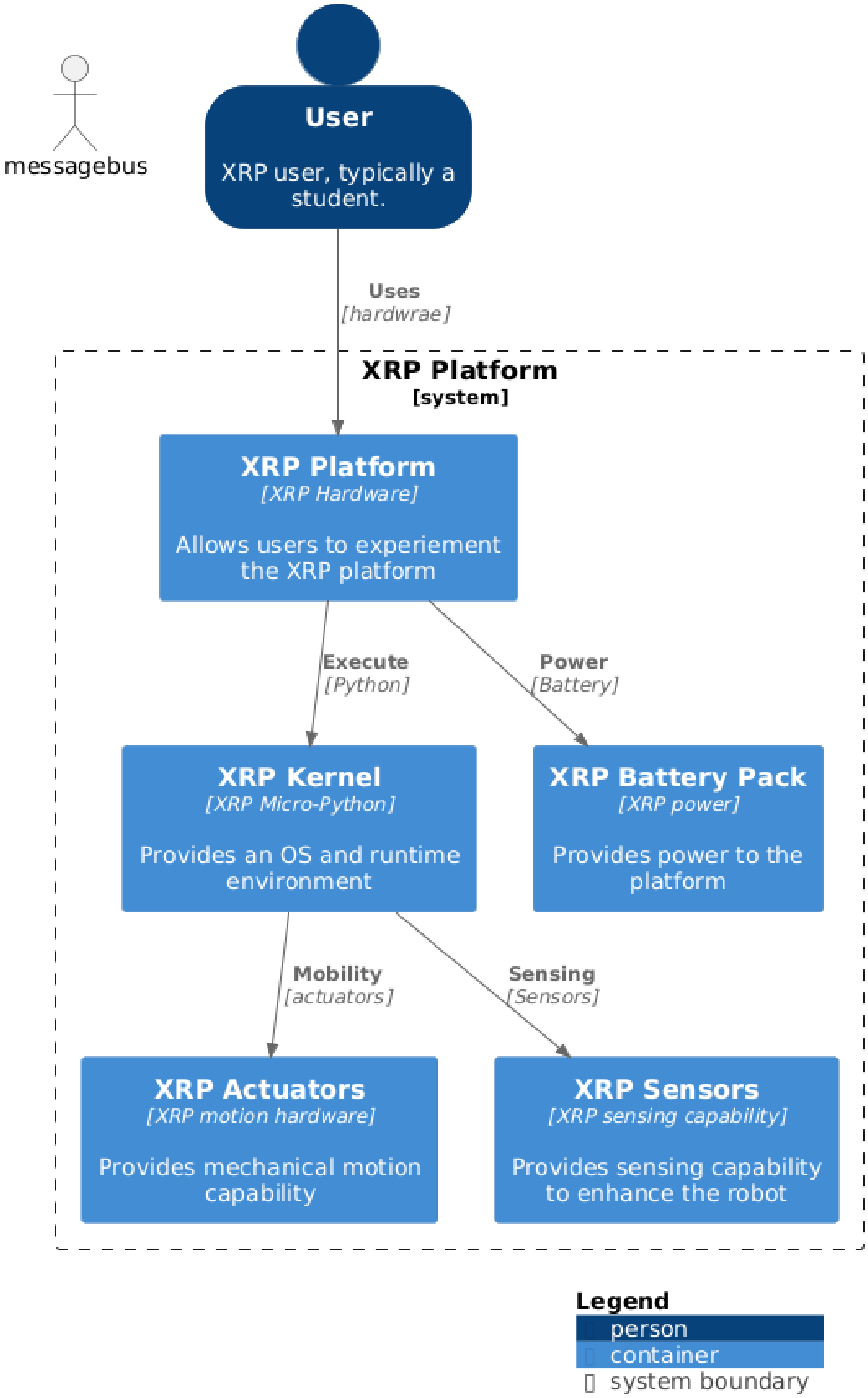

XRP
[enterprise]
User
XRP user, typically a student.
Asks questions to
[Email, Discord]
XRP Admin
XRP Web Administrator
enterprise
student run software on the XRP robot
develope software for the XRP.
manage XRP Web deployment and maintainance of XRPWeb
XRP Robot
XRP Hardware platform.
Update XRP or Run the robot program
XRP Web App
Allow XRP users to write robotics program for the XRP on the XRP website.
Legend
person
system
external person
external system

Containers

The XRP hardware platform is a afordable hardware kit that consists of the following components. The kit is retail for $114.95. Sparkfun.com offers discount to FIRST teams.

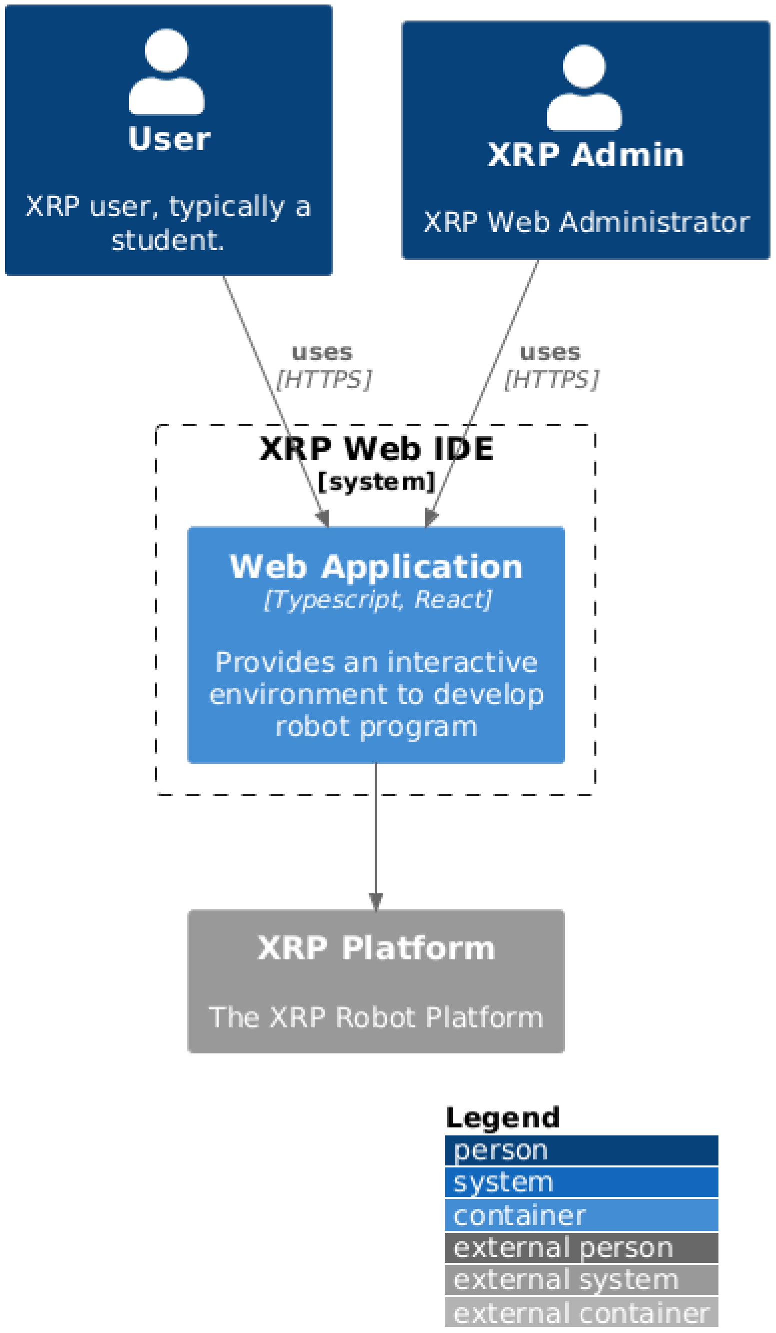
- Sparkfun controller board
- XRP chassis set
- Ultrasounic Distance Sensor
- Servo (sub-micro size)
- Hobby Motor with encoder

- Caster wheel
- O-Ring
- flexible Qwiic Cable - Female Jumper (4-pin) with Heat Shrink
- Battery holder
- Custom Line Follower board
- XRP Sticker Sheet


messagebus
User
XRP user, typically a student.
Uses
[hardwrae]
XRP Platform
[system]
XRP Platform
[XRP Hardware]
Allows users to experiement the XRP platform
Execute
[Python]
Power
[Battery]
XRP Kernel
[XRP Micro-Python]
Provides an OS and runtime environment
XRP Battery Pack
[XRP power]
Provides power to the platform
Mobility
[actuators]
Sensing
[Sensors]
XRP Actuators
[XRP motion hardware]
Provides mechanical motion capability
XRP Sensors
[XRP sensing capability]
Provides sensing capability to enhance the robot
Legend
person
container
system boundary

The XRP software platform (XRP Web development environment) provides the students environment where they can write robot program in block or Python programs.

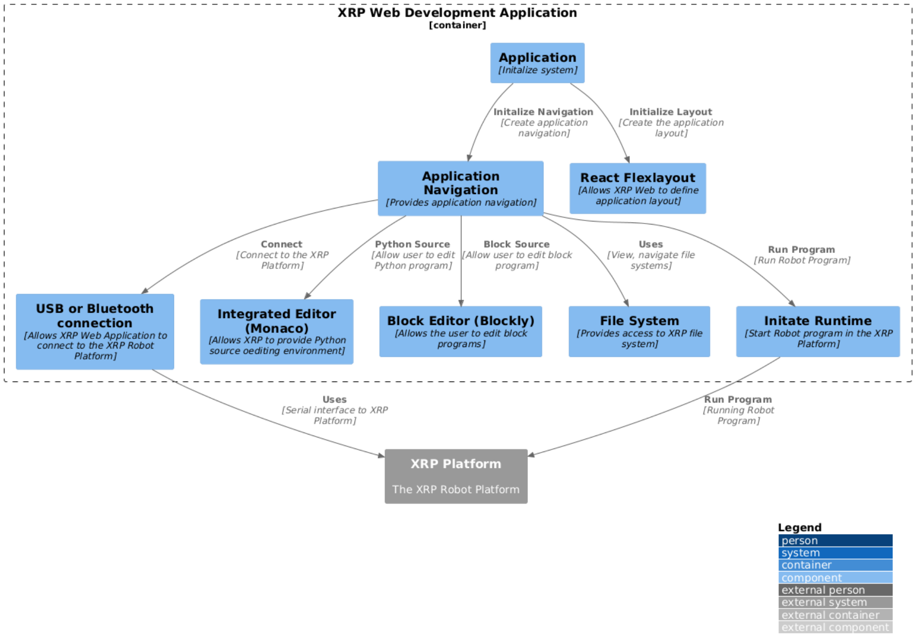
Container diagram for XRP Web



Components

The software components are the major features of the system where they are grouped distintively and their relationship between components are specified. Each component can be developed independently and each have interface specification between them.

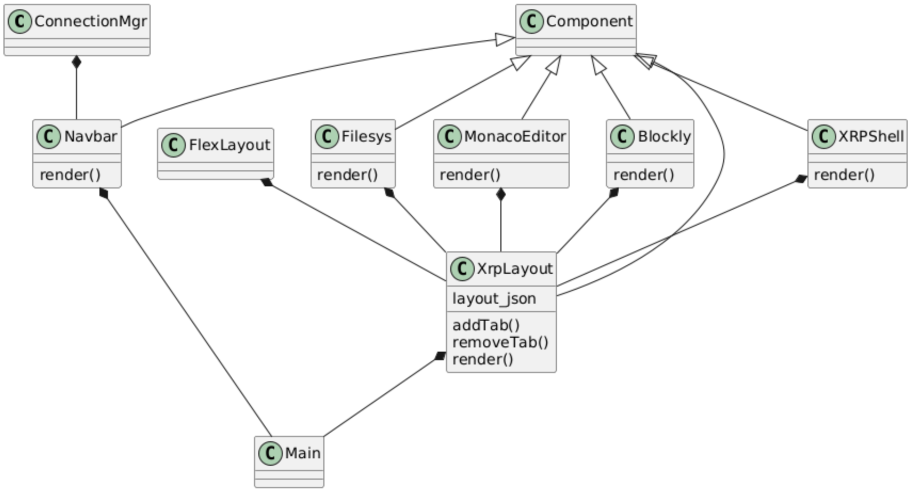
Component diagram for XRP Web Development Application


XRP Web Development Application
[container]
Application
[Initalize system]
Initalize Navigation
[Create application navigation]
Initialize Layout
[Create the application layout]
Application Navigation
[Provides application navigation]
React Flexlayout
[Allows XRP Web to define application layout]
Connect
[Connect to the XRP Platform]
Python Source
[Allow user to edit Python program]
Block Source
[Allow user to edit block program]
Uses
[View, navigate file systems]
Run Program
[Run Robot Program]
USB or Bluetooth connection
[Allows XRP Web Application to connect to the XRP Robot Platform]
Integrated Editor (Monaco)
[Allows XRP to provide Python source oediting environment]
Block Editor (Blockly)
[Allows the user to edit block programs]
File System
[Provides access to XRP file system]
Initate Runtime
[Start Robot program in the XRP Platform]
Uses
[Serial interface to XRP Platform]
Run Program
[Running Robot Program]
XRP Platform
The XRP Robot Platform
Legend
person
system
container
component
external person
external system
external container
external component

Codes

This section documents the major classes which are based on the React Component class.

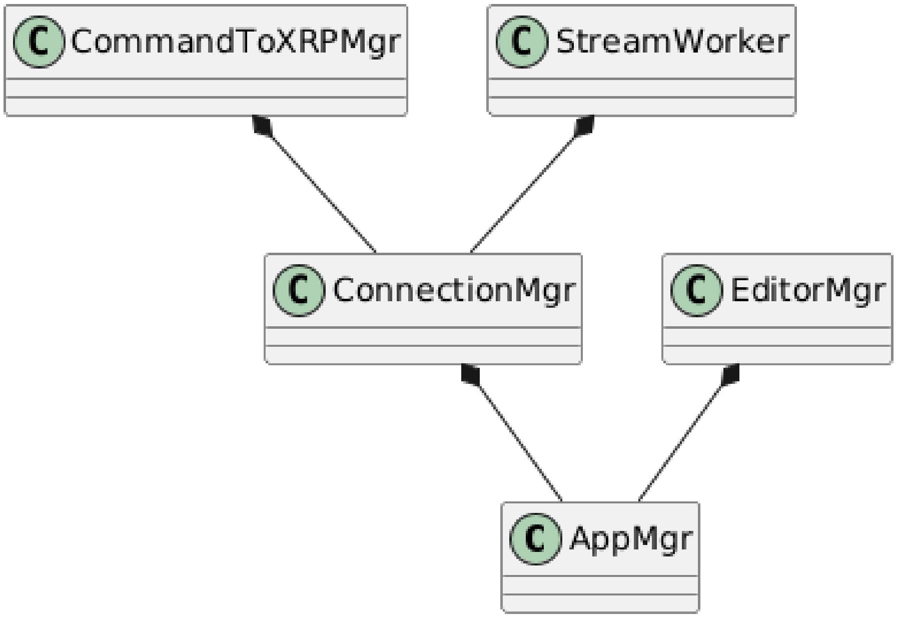
React Components



Application Components

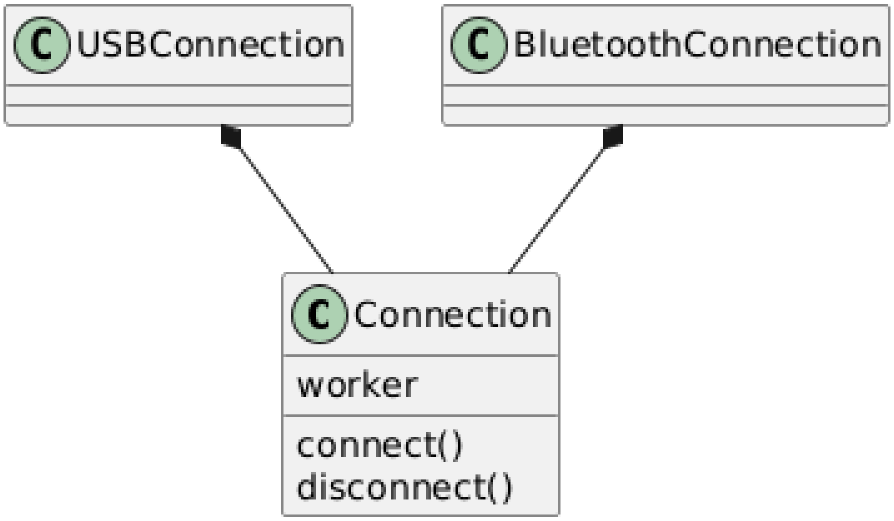
The application components manages the core functionality of the following features.

- Application Management
- Connection Management (Cable and Bluetooth)
- File Management (User and System Filesystem)
- Editor Session Management



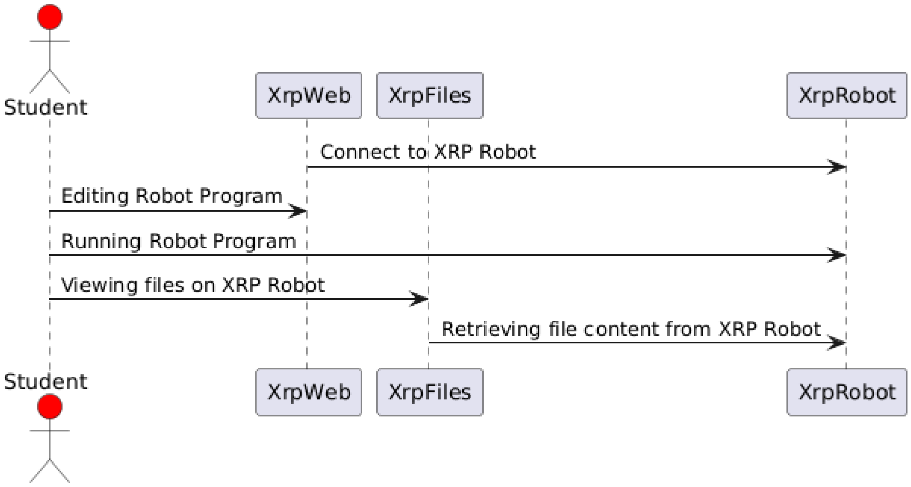
Connections

There are two type of connections in the system, i.e., USB or Bluetooth. The connection classes provide interfaces to connect or disconnect and handle the communication between the Web Application and the XRP Robot. Once the connection is established, data transfer from the connection objects to file system manager is handled through publish and subscribe interfaces.



Run Time View

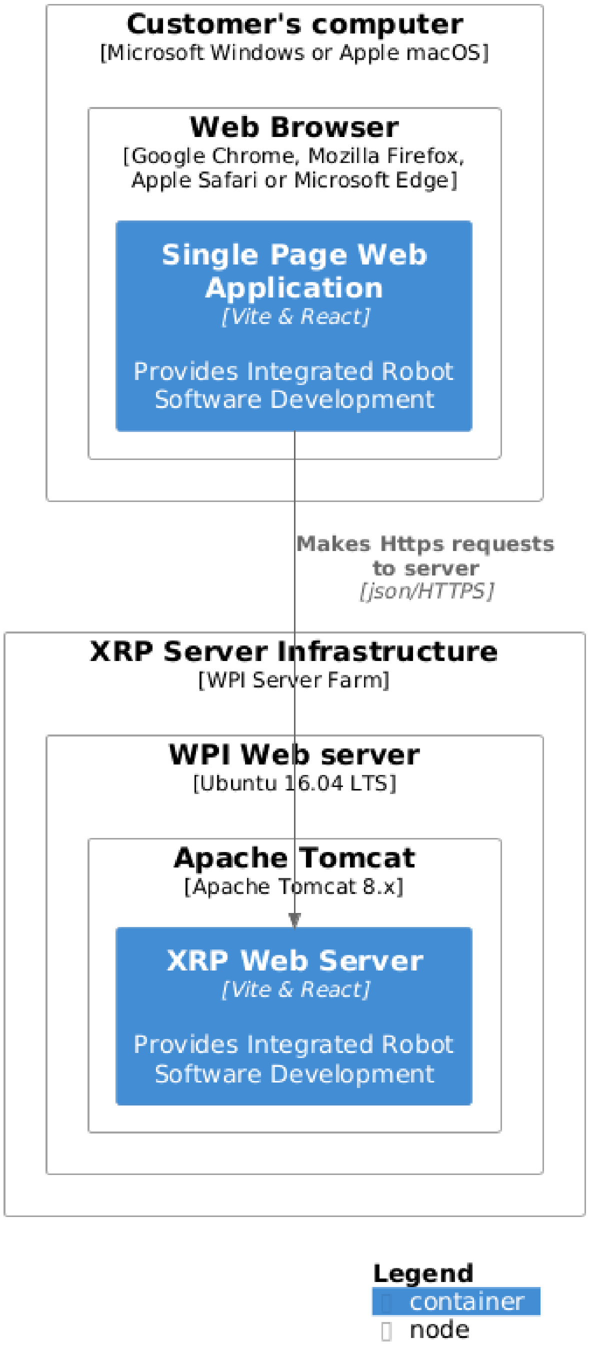
The figure below depicts the runtime view of the system when the users are interacting with the system.



Deployment View

The figure below depicts the how the Web application is deployed.

Deployment Diagram for XRP Web Development Application


Customer's computer
[Microsoft Windows or Apple macOS]
Web Browser
[Google Chrome, Mozilla Firefox, Apple Safari or Microsoft Edge]
Single Page Web Application
[Vite & React]
Provides Integrated Robot Software Development
Makes Https requests to server
[json/HTTPS]
XRP Server Infrastructure
[WPI Server Farm]
WPI Web server
[Ubuntu 16.04 LTS]
Apache Tomcat
[Apache Tomcat 8.x]
XRP Web Server
[Vite & React]
Provides Integrated Robot Software Development
Legend
container
node

Architecture Decisions

Vite and React was choosen as the user interface development framework as they are supported by the Open Source community as well as companies like Facebook and Vercel. Both are respected UI framework which are used by a large development comminity.

Quality Requirements

The XRP Web Development application is developed and meet the following quality requirements.

- Unit Tests
- End to End Test

Security Requirements

XRP Web Development application is hosted in a server farm at WPI. Https secured protocol is used. A valid server HTTP server certificate is configured for the XRPCode subdomain.

Coding Standards

Every organization has its own software development standard. The following is some of the recommend best practices.

Naming Convention

The goal of of having a standard naming convention in the code base is to provide the following attributes of software engineering principles.

- Readability
- Maintainability
- Organization
- Communication

These attributes help create a cohesive and structure codebase that is easier to work with and to help reduce errors and to promote collaboration among developers. Readability and Maintainability attributes contribute the overal software development styles and consistency in the codebase. The last two attributes create a collaborative environment and froster transparency and improve healthy organization interactions.

The following list some popular naming conventions.

- PascalCase
- camelCase
- kebab-case
- SCREAMING_SNAKE_CASE

Pascal Case Pascal Case refers to the convention of writing compound words in which the first letter of each word is capitalized. There is no space or punctuation between words. The typical use case is class name or component name. See the below example.

```
// React Component
const NewFile = () => {
    // ...
}

// Enumeration
export enum ConnectionType {
    USB,
    BLUETOOTH
}

// types
export type ListItem {
    label: string,
    image: string
}
```

camelCase Camel Case refers to the convention of writing compound words or phrases where each word begin with a capital letter except the first word. This convention always starts with a lower case letter of the first word.

```
// Variable Name
const [filename, setFileName] = useState('')

// Function Name
const getFilename = () {
    return this.filename;
}

// Object properties
const userOptions: ListItem[] = [
    {
        label: 'Kevin',
        image: blocklyIcon,
    },
    {
        label: 'Frank',
        image: pythonIcon,
    },
    {
        label: 'Kathy',
        image: pythonIcon,
    },
];

// Custom Hooks
const useFile = () => {
    // ...
}
```

Kebab Case Kebab case refers to the convention of writing compound words in lowercase name with a hyphens (“-”) between the words. Kebab case is popular for using in CSS class names. Since this architecture specified Tailwind CSS as the CSS utility framework, all of the Tailwind CSS styling are specified inline with CSS utility classes. Kebab case is used sparingly. However, this convention can be used in naming source files.

Folder Structure

A clear and organized folder structure provides easy navigation of the source tree and grouping of all related components in the subtree. A well-designed folder structure is essential for code maintainance and for codebase scalability. An example of the folder structure is shown below.

```
src/                    // Root folder for the source code
├── features/        // Grouping features of the application
│   └── ...        // Other feature folders (e.g., authentication, dashboard)
├── components/          // Reusable components
│   ├── ...        // Component folders (e.g., Button, Header)
├── utils/               // Utility functions
│   ├── ...        // Helper functions or services (e.g., dateUtils, apiUtils)
├── assets/              // Storing static assets
│   ├── images/     // Storing image files (e.g., logos, icons)
│   ├── fonts/      // Storing font files (e.g., custom fonts)
│   └── ...        // Other asset types (e.g., videos, audio)
├── styles/              // Global styles
│   └── ...        // Style files (e.g., global.css, themes)
```

```
├── App.tsx              // Entry point of the application
└── ...            // Other necessary files and folders (e.g., index.tsx, routes.ts
```

Development Environment

This software architecture is based on all open-source software and tools. It leverages existing React components publish via npm. Visual Studio Code (VSCode) is the recommended developer integrated development environment (IDE). This IDE provides many helpful extensions to ease software development. Github repository is used to maintain the codebase.

Recommended VSCode plug-ins

- ESLint
- React Extension Pack
- Tailwind CSS Intellisense
- Vitest
- Git Graph
- PlantUML
- Prettier
- Github Copilot Chat

Ricks

The following list a potential list of risks.

- Open Source software library packages are out of date and their associate repositories are lack of maintainance.
- Lack of future volunteer developers to support the project