Technology stack – Architecture Choices

# Introduction

One of the major things about the smart hydro project is that there are many constraints that we must work with as well as many things to cater for at the same time. For this reason, we must choose technologies that will synergize very well so that the system can offer accurate live data of the hydroponic tent, correctly fetching the data from the tent like sensor readings, and relaying it back to the clients app. For this many decisions were made.

We don’t only value what works best for the team, but more specifically what works best performance wise and accessibility wise for the farmers. This is because we would have to cater for live data and real time communication while also accounting for those that don’t have access to internet connections. The choices made will be made on what fits for the farmers as well as the team’s creativity and ability to deliver.

# Architecture choices

## Cross-platform development

One of the major things that stood out to us would be what platform we would use to develop the application.

Initially we wanted to make an Android only application as that aligns with our studies in Varsity College, however we quickly came to realise that does not cater for everyone. After this we had a look at the 2 next possible options for cross platform:

* React Native
* Flutter

Each of these are tried and tested frameworks that will allow developers to create beautiful cross-platform apps. However, there are some drawbacks with the 2 for the team.

This is extremely powerful and allows for a shared look and feel across Android and iOS however the downside of this for the development team is the expert skill in JavaScript. While the team would be able to pick up the skills required for this, it would be time consuming and that time could be used for other parts of development.  
Getting to know the specifics for each code base would be a challenge for time if the team wanted to do create the best possible native application.   
There is a newer 3rd option that the team would be more comfortable with.

### Kotlin-Multiplatform

This resonates best with the team as the entire development process would be done with Kotlin, and it also utilizes shared code bases. This will ultimately help the development of the team.   
With this it will allow us to code in an environment that we are used to as well as making use of Compose Multiplatform which is the cross-platform UI toolkit that will make this possible. This will create stunning visuals and since the team already has exposure to this, it works in their favour. This will be key in creating a native application for the farmers and users who want a more in-depth view of the application. Offering native response time and feedback.

With its powerful shared code base that allow the creation of business logic to be shared across all platforms, this makes this a top contender for the team’s final choice.

### Consideration for the target audience

Creating an offline first application is crucial as that will be for the core users that are part of the joint research for Smart Hydro. This means that when doing so we would have to accommodate for core functionality to be done offline and if there is internet access, it should be able to be used completely without internet access, present local data immediately, and fetch data in a manner that does not eat away at the battery and storage.

### What does this mean for the client and farmers?

Well for starters we would want to target both platforms irrespective of what the farmers use. This could cater for mobile applications running on android or running on iOS. We also aim to create a desktop application as an add on to this as no one should be restricted to just mobile use.

Unlike react native, KMP compiles code for all different target platforms to completely native code. This means that you would get the same feel and performance as if the iOS app were to be developed in Xcode via Swift and objective-c (Kotlin Multiplatform, 2025). This is key as we would want the farmers to have a very smooth look and feel to the application regardless of how old their devices are or what platform they are on. Hence this would be a key part that we decide when picking a final technology.

## Backend framework

For this there were 2 main options that appealed to the team

* Express.js
* Ktor

These choices are very crucial to the system design however we have opted to go for express.js.

### Ktor

#### Pros

1. Lightweight and modular
2. Kotlin-centric
3. Easy to learn as there are many documents
4. High performance due to Kotlin’s coroutines. (Chaewonkong, 2023)

#### Cons

1. Smaller ecosystem - their community is still growing, and library support is not as extensive as other frameworks out there.
2. Less mature – Ktor is still a new framework, and some features and best practices are still evolving. (Chaewonkong, 2023)

### Express js + Node js

#### Pros

* Minimal and flexible like Ktor
* Also, very fast
* Has good middleware – built-in features like logging and authentication
* Has a more extensive ecosystem

#### Cons

* Relatively steeper learning curves
* Express js is unopinionated, meaning it may require additional making on project structure. (GeeksforGeeks, 2024)

Due to the more extensive support from the JavaScript community, express js seems like a more powerful option and with libraries like socket.io this allows for easy integration of web sockets, allowing for live data communication. To clarify with this Node js is a runtime that would allow for the backend capability and express js would be the framework on top of that that allows for easy routing and reducing some more complex configurations.

### What does this mean for the client and farmers?

In relation to the frameworks, this is something not to relevant to the farmers who don’t have internet access, but for those who do and want to have access to remote monitoring, speed and reliability is crucial. This means that the choice would be in the odds of the frameworks that are more extensively supported as future maintenance, and the lifecycle is very important to keep this going.

## Frontend architecture choice

For this we were also with a choice of 2 things, namely React and Next.js

### React

React is an incredibly popular JavaScript framework that allows to create beautiful front-end applications for the web. (Masters, 2023)

#### Pros

1. Highly efficient by updating and rendering the UI in web applications using a virtual DOM (Document Object Model). This is basically creating a virtual DOM whereas using vanilla JavaScript (pure JavaScript with no framework) interacts with the native Dom. This may be a little more computationally expensive working with a virtual layer however this will only be for more complex UI frontends which ours will not be as in depth.
2. Reusable components – like how Jetpack compose makes use of reusable code components, we can code a component once, for example a button, then call that design in multiple pages of the client app giving a consistent look and quicker development.
3. SEO-friendly - meaning it will more likely be recognized by search engines like Google and Bing. This allows for the site to be more likely appear in search engine results. (Masters, 2023)

#### Cons

1. JSX – JavaScript xml, it can be a little confusing to those unfamiliar to react development
2. Component-based architecture – meaning UI is made via the UI components but it can be challenging for beginners to grasp initially.
3. State and props – managing state for the components so that it displays the correct information based on user interactions. (Masters, 2023)
4. Component Lifecyle – making sure that the components are triggered at the correct point In time and terminate when they should be another key part.
5. Updates and changes - react frequently receives updates and staying up to date with best practices can be tedious. (Masters, 2023)

### Next js

This can be seen as an extension of React with more features and functionality, essentially adding onto the good features that react has.

#### Pros

1. Improvised Performance with server components – enhances performance by rendering components on the server side which reduces the resources needed on the client side to render UI allowing for faster load times for the client side making it feel more responsive.
2. Automatic image optimization - the framework automatically optimizes the images by choosing suitable formats and resolution based on the user’s device and browser.
3. Enhanced developer experience with faster refresh - Next js 13 has near instantaneous refresh times allowing for faster development and uses a rust-based system, called Turbopack (Stephen, 2023). Turbopack is an incremental bundler optimized for JavaScript and TypeScript, written in Rust, and built into Next.js. You can use Turbopack with both the Pages and App Router for a much faster local development experience (Vercel, 2025).
4. Built in middleware support

#### Disadvantages

1. Learning curve for server components
2. Potential overhead for smaller project
3. Limited community and Third-party support.

What we will add to this decision is, the main developer of the project has some basic experience with jetpack compose and understands the basics of life cycle aware components, UI state and state management. So, these work to the strengths of the development team, the only thing that will be needed to be done is transfer this understanding and apply it in JavaScript and the appropriate frameworks.

## Arduino + Raspberry pi

This is working of the existing code bases to which we will add onto it and configure the appropriate server Api endpoints so that the data can be sent to and from the server to the hardware. This will be done via MQTT where we work with a light weight publish-subscribe protocol that is specifically designed for IoT. This will be unchanged as this has been in development over the years and is in good working order.

Convolutional Neural Network (CNN) for On-Device Facial Recognition

The primary purpose of this CNN model is to enable on-device facial recognition for the native applications. For this the one option that stands out to us is TensorFlow that has extensive community support and is very well documented. There is also plenty of documentation and resources available to work with. With this we will be able to train our own model for the farmers is generally done by the post graduate students. However, the lead developer has taken the task of creating a Convolutional Neural Network. Neural networks are a subset of machine learning and the foundation of deep learning algorithms. They are made up of node layers, which include an input layer, one or more hidden layers, and an output layer. Each node connects to another and has a weight and threshold. If the output of any individual node exceeds the specified threshold value, that node is activated and sends data to the next layer of the network. Otherwise, no data is transmitted to the next layer of the network. (IBM, 2021)

### Use case

This CNN model will be used to identify the farmers by face, and it will run locally on the mobile device. This will be done with a tensorflowlite file and adjust the UI and notifications based on the farmer. For example, if the app recognizes the farmer, it will then tailor it to their preferences and their access to internet. Meaning if there are certain features that purely rely on internet access, it will not have these available to these users.

### Technical Implementation

Training data will be collected from the farmers. This will be in the form of videos/images in different lighting, different settings/backgrounds to make sure that there is enough data to work with for the data set.

For labelling, we will label these based on the farmers identity, so who they are specifically and what their preferences are.

For inference of how we will be making use of this TensorFlow model, we will use TensorFlow Lite to call this on the native application ensuring that this can work locally with the farmers even if they are offline.

### Ethical and practical consideration

We will collect consent from the farmers for this, ensuring that they allow for this to be done as we will be using images/videos of their face and would not want to make them feel uncomfortable.

If the camera fails or does not work, we can gracefully go to the default app look and feel which will still be generalized for the farmers. This is because at most the CNN model will change the notifications by adding in their name for sending notifications to the farmer about the system, as well as the UI to their preference like colours or even Wi-Fi capabilities.

For the accuracy of the model, this will be incrementally tested making sure that we have as little loss as possible when calculating the actual output of the model. This means that it will not be a one iteration build but rather built in increments tying to the scrum development methodology where we aim to incrementally build the system.

### Client considerations

This was specifically mentioned by the client, Sarina Till, who thought that it would be a great bonus factor to add to the system so that it could make the application feel more immersive for the users, allowing them to have the application view who the farmer is (model will be trained on a defined set of farmers) and alter the UI based on who they are to their preferences and their visible emotion on their face. This would be a powerful addition that also adds into user experience for the farmers making them feel more comfortable.

## Koog AI

This will be an AI Agent that will be able to allow us to automate the system even further on the client-side application by automatically generating summaries for the user as well as automate tasks when things go wrong in the tent. For example, if the tent has readings that are not in line and may disrupt the growth of the plants, the AI agent will assess this and according to the sensor and trigger the appropriate actions within the tent.

This will be useful for the farmers that have Wi-Fi connectivity as this will automatically trigger what needs to be done. This will ensure that the farmer must do as little as possible while allowing for them to monitor their plants and keep track of them. This is a Kotlin based AI framework that has been released recently and is perfect for working with the Kotlin multiplatform application (Koog.ai, 2025).

## Database services

### Local Database

Since the application choosing the correct local database is crucial

Reference List

Reference list

Chaewonkong, 2023. *Ktor: The Next Generation Framework That Might Replace Spring Boot*. [online] Medium. Available at: <https://medium.com/@chaewonkong/ktor-the-next-generation-framework-that-might-replace-spring-boot-868e8d21fc0f> [Accessed 25 June 2025].

GeeksforGeeks, 2024. *What is Express?* [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/node-js/what-is-express/> [Accessed 26 June 2025].

IBM, 2021. *Convolutional Neural Networks*. [online] Ibm.com. Available at: <https://www.ibm.com/think/topics/convolutional-neural-networks> [Accessed 26 June 2025].

Koog, 2025. *Overview - Koog*. [online] Koog.ai. Available at: <https://docs.koog.ai/> [Accessed 26 June 2025].

Kotlin Multiplatform, 2025. *Kotlin Multiplatform Help*. [online] Kotlin Multiplatform Help. Available at: <https://www.jetbrains.com/help/kotlin-multiplatform-dev/multiplatform-configure-compilations.html> [Accessed 26 June 2025].

Masters, R., 2023. *Advantages and Disadvantages of React JS - React Masters - Medium*. [online] Medium. Available at: <https://medium.com/@reactmasters.in/advantages-and-disadvantages-of-react-js-e6c80b25763b> [Accessed 26 June 2025].

Stephen, 2023. *Next.js — Advantages and Disadvantages over React - Stephen - Medium*. [online] Medium. Available at: <https://weber-stephen.medium.com/next-js-advantages-and-disadvantages-over-react-c87aa20de64f> [Accessed 26 June 2025].

Vercel, 2025. *API Reference: Turbopack*. [online] Nextjs.org. Available at: <https://nextjs.org/docs/app/api-reference/turbopack> [Accessed 26 June 2025].