**World Hunger**

Urban Farming Solutions

By

Micail Bakhshzaad

Submitted to

**The University of Roehampton**

In partial fulfilment of the requirements

for the degree of

**BACHELOR OF SCIENCE IN COMPUTING**

A black line drawing of a butterfly

AI-generated content may be incorrect.

**Date:** 07/08/2025

Signed

**Enter your name here**

**Micail Bakhshzaad**

**Declaration**

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Acknowledgements

Lisa Haskel

Abstract

In cities with little agricultural land, urban food insecurity is still a serious problem. Through this effort, Urban Farming Solutions was created, a website that provides easily available instruction and community resource sharing to enable urban residents to grow their own food.

The literature review looked at vertical farming systems and community gardens as urban farming methods. Examining sources such as studies on community-based agriculture, FAO reports, and research on climate adaption. Although there were technical fixes available, the evaluation found notable limitations in beginner-friendly, easily available materials for future city farmers.

The HTML/CSS/JavaScript frontend, MongoDB database, and Node.js/Express backend were used in the implementation. Strict sprints had to give way to a flexible development approach due to health restrictions. 12 growth guides with AI-generated videos, secure authentication, community resource sharing for seeds and tools, and weather API connection for location-based recommendations as well as a crop progress tracker are all included in the finished product.

Strong usage was demonstrated during testing with 8 participants, as seen by 3.5/5 satisfaction ratings and 100% job completion for main functionalities.

Despite lacking intended features like the community forum, the project effectively demonstrates that straightforward, Digital tools that are well-designed can encourage the use of urban gardening. By encouraging resource sharing and making growing information easily available, The platform provides a useful strategy for tackling food security through community development and personal empowerment. According to the review, even simple solutions that are user-focused can have a significant impact on reducing urban hunger and promoting sustainable food production in urban areas.

Table of Contents

[1. Introduction **Error! Bookmark not defined.**](#_Toc927214740)

[Research Question or Problem statement **Error! Bookmark not defined.**](#_Toc522749298)

[Aims vi](#_Toc1298644285)

[Objectives vi](#_Toc198637622)

[Legal, Social, Ethical and Professional Considerations vi](#_Toc960797214)

[Background vii](#_Toc273743648)

[Report overview vii](#_Toc210207254)

[2. Literature or Technology Review viii](#_Toc1335231035)

[3. Methodology xiii](#_Toc1844834773)

[4. Implementation xvi](#_Toc1739317238)

[5. Results xvii](#_Toc826321870)

[Evaluation **Error! Bookmark not defined.**](#_Toc1085198293)

[Related Work **Error! Bookmark not defined.**](#_Toc55229855)

[6. Conclusion xviii](#_Toc253686570)

[Reflection xviii](#_Toc1734103406)

[Future Work xviii](#_Toc542248490)

[7. References xix](#_Toc1743583894)

[8. Appendices xx](#_Toc649705763)

# **Introduction**

This project promotes urban farming as a workable solution to the problem of food insecurity. Building a website that instructs users on how to cultivate and harvest crops in various climates is the aim, with an application that promotes resource sharing within a community. This project is crucial because it gives people the tools, they need to produce food on their own, which lessens the effects of hunger.

## **Problem Statement**

**What is the problem being addressed?** Hunger is still a major worldwide concern, millions of people are insufficiently fed. Because of limited agricultural territory, urban regions have unique challenges in terms of food security.

**Who is affected by the problem?** The problem impacts urban populations, especially those living in food deserts with limited access to fresh vegetables**.** It also affects low-income families who struggle to purchase nutritious food.

**Why is it important to solve the problem?** Reducing poverty, enhancing public health, and encouraging sustainable urban growth all depend on addressing world hunger. Public education about urban farming can help reduce food insecurity by increasing access to fresh produce.

## **Aims**

1. To increase understanding of the worldwide problem of world hunger and how it affects urban areas.
2. To educate people about sustainable urban agricultural strategies appropriate for varied climates.
3. To build a platform that promotes the sharing of urban farming resources and knowledge.

## **Objectives**

1. Create a user-friendly website with detailed instructions on growing and harvesting crops in various climatic situations.
2. Make an application that enables community members to exchange farming supplies like seeds, tools, and containers.
3. Incorporate a forum on interactive urban agricultural architecture that showcases creative concepts and best techniques.

## **Legal**

The project will make sure that data protection rules are followed, especially while processing user data on the website and application. Regarding any shared materials or content, intellectual property rights shall also be upheld.

## **Social**

By fostering cooperation and resource sharing among urban dwellers, the project will increase community involvement. To combat food insecurity, it seeks to promote a sense of shared responsibility.

## **Ethical**

By making sure that all the information supplied is correct and helpful, the project will solve ethical issues. Additionally, it will guarantee that a wide range of user groups can access and utilize the platform. Personal information will be safeguarded and kept private by handling user data with the highest care. To ensure user privacy, the project will abide by data protection laws like the GDPR (General Data Protection Regulation). Actions will be taken immediately in the event of any unethical behaviour or data breaches. include investigation and resolution, to maintain the websites integrity.

## **Professional**

Continuous monitoring: putting in place feedback systems to improve the platform continuously.

User Centric Design: putting a lot of focus on user experience and usability when designing the application and website.

## **Background**

Urban farming, which uses accessible city spaces for food production, is an innovative approach to fight world hunger. Urban agriculture has been shown to greatly increase food security and resilience in urban areas. To develop an engaging and instructional platform, this project integrates both the most recent research and fundamental topics in sustainable agriculture and urban planning. The website seeks to give readers a comprehensive grasp of urban agricultural techniques and their effects on food security by fusing new research with well-established knowledge.

## **Report overview**

The report will cover the following structure:

**Literature Review:**

* a review of current studies on urban farming and how it affects food security.

**Methodology:**

* An explanation of the steps involved in developing websites and applications.

**Evaluation/Results:**

* full evaluation of the project's effects on food security and urban farming techniques.
* results of user testing and feedback regarding the website.

**Conclusion:**

* An overview of the results and suggestions for more research.

# **Literature - Technology Review**

## **Literature Review**

**Introduction to World Hunger and Urban Farming:**

Millions of people experience food insecurity, making world hunger a serious worldwide problem. By using existing city spaces for food production, urban farming has drawn attention as a potential remedy for this dilemma. Numerous important institutions and experts have investigated this idea, greatly advancing and comprehending it.

**Problem Description**

Numerous projects concentrating on sustainable solutions have been prompted by the growing frequency of food insecurity. Promoting urban farming as a workable solution to food insecurity is the goal of this project. The objective is to develop an application that encourages a community for resource sharing in addition to a website that instructs users on how to produce and harvest crops in different climates.

The following tools are used in this project:

* **Frontend**: HTML, CSS, JavaScript - For responsive and interactive, designed to provide an engaging experience.
* **Backend**: Node.js, Express - will control server-side logic, allowing to produce dynamic, climate-specific guidance.
* **Database**: SQL – will enable effective data storage and management, assuring consistent handling of user information and content.

The project's relevance stems from allowing individuals to achieve independence in food production, thereby mitigating the effects of hunger.

**Key Literature and Contributions**

**Urban agriculture and environmental sustainability,** <https://www.researchgate.net/publication/370000029_Urban_agriculture_and_environmental_sustainability>: This article explores urban agriculture's environmental sustainability, which is in line with the project's emphasis on sustainable methods. It provides proof that urban farming can reduce the environmental impact of food production. This contributes to the project's teaching aspect by highlighting the environmental benefits of urban farming.

**Global Trends in Urban Agriculture Research: A Pathway toward Urban Resilience and Sustainability,** <https://www.mdpi.com/2073-445X/11/1/117>: Sustainable land use, which is crucial for urban agriculture, is covered in this article. It promotes urban farming, which is the project's primary goal, and offers insightful information about how to use urban areas for food production. The study emphasizes the advantages of urban agriculture for the environment and space, highlighting its potential as a means of addressing food poverty.

**The Food and Agriculture Organization (FAO),** <https://www.fao.org/home/en>: offers a wide range of information about sustainable agriculture and global food security. This website provides case studies and reliable statistics about how urban farming can help combat food insecurity. The FAO's rules and reports form the basis of the project's website's instructional content.

**AGROPOLIS: THE ROLE OF URBAN AGRICULTURE IN ADDRESSING FOOD INSECURITY IN DEVELOPING CITIES,** <https://www.jstor.org/stable/24461745#:~:text=13%20Greater%20food%20availability%20and,both%20to%20low%2Dincome%20households.&text=agricultural%20goods%20they%20produced.,urban%20poor%20in%20developing%20cities>: This article examines how community-based urban agriculture may advance social justice and food justice. The article emphasizes the significance of community participation in urban farming projects, making it especially pertinent to the project. in line with the project's goal of creating a community for the exchange of resources. A crucial component of the project's strategy, the piece also emphasizes how urban farming may strengthen local communities.

**Urban Agriculture Food, Jobs and Sustainable Cities,** <http://www.jacsmit.com/book/Chap02.pdf>: This chapter of the book explores the idea of urban farming as a means of addressing urban food insecurity. It highlights how urban agriculture can be used to produce food in underutilized urban areas. It supports the project's objective of teaching people how to grow crops in a variety of climates. The resource is useful for giving basic information about how urban farming and food security are related.

**Vertical Farms: From Vision to Reality**, <https://news.climate.columbia.edu/2011/10/13/vertical-farms-from-vision-to-reality/>: The idea of vertical farming, a high-tech urban farming technique intended to optimize space in urban settings, is explored in this article. Since it provides information about cutting-edge agricultural technologies that may be included into the project's website, it is beneficial to the project. The resource is especially useful for offering substitutes for urban farming in constrained urban areas.

**Community-based urban agriculture for food justice: a review,** <https://www.researchgate.net/publication/366920880_Community-based_urban_agriculture_for_food_justice_a_review>: The contribution of community-based urban agriculture to the advancement of social justice and food justice is studied in this article. This article highlights the ways in which urban farming may strengthen local communities.

**Summary of Main Points**

* Food security in urban areas can be greatly improved by urban farming.
* The effectiveness of urban agriculture depends on its integration with city planning.
* Urban farming methods can become more efficient with the help of technological advancements like vertical farming.

**Discussion**

All the evidence points to urban farming as a viable way to address food poverty. However, there are obstacles to overcome, like the expensive initial setup expenses and the requirement for technical know-how. There are clear benefits and drawbacks to the various strategies, which range from high-tech vertical farming to conventional urban farming.

**Critical Evaluation**

Though limited by climate and area, traditional urban gardening is affordable and accessible. On the other hand, vertical farming gives a high yield but necessitates technical expertise and a significant financial commitment. The literature offers a nuanced perspective, implying that tailored solutions are required for various urban environments.

**Application to the Problem**

The project will use the knowledge gathered from the literature study to inform the creation of an application and website aimed at helping and educating urban residents. The project's implementation strategy will be shaped by the focus on sustainable practices and community involvement. making sure it considers the unique requirements of urban communities that are experiencing food insecurity.

## **Technology Review**

Frontend Technology:

* **HTML, CSS, JavaScript**: For the instructional website, these technologies will be utilized to develop an interactive and responsive user experience. The material will be structured by HTML, styled by CSS, and interactive by JavaScript.

Backend Technology:

* **Node.js, Express**: The Express framework, a JavaScript runtime, and Node.js, will be employed to manage the making of dynamic content and server-side logic. The development of climate-specific guides will be made easier by these technologies, which will also manage user interactions with ease.

Database Technology:

* SQL Server: To effectively store and handle data, a SQL database will be used. The organized layout will make it simple to get and modify user data. information on resource sharing and content for guides tailored to a particular climate.

**Reason for Technology Choices**

Frontend: Users can access and enjoy the website thanks to these technologies. As they are widely used, simple to use, and capable of producing a dynamic user experience.

Backend: chosen because to its flexibility and effectiveness in managing simultaneous tasks, which are necessary for managing dynamic material and conducting user interaction.

Database: SQL is used because of its reliability, the convenience of data management, and good support for complicated queries. This makes it appropriate for handling the project's data requirements.

## **Summary of Outcomes of Literature and Technology Review**

|  |  |  |
| --- | --- | --- |
| Article | Benefits | Limitations |
| Urban Agriculture and Environmental Sustainability | demonstrates how urban farming, in line with sustainable practices, can lessen the environmental impact of food production. | highlights the environmental component first and foremost, this might not cover all urban farming's social components. |
| Global Trends in Urban Agriculture Research (MDPI) | highlights how urban areas may be used for food production and supports urban farming as a sustainable land-use strategy, matching the project's objective of teaching diverse farming techniques in various climates. | It's likely that the article lacks thorough, realistic implementation strategies for various climates. |
| The Food and Agriculture Organization (FAO) (FAO.org) | supports the importance of urban farming in ensuring food security by offering case studies, research, and international statistics. The website's educational material can be informed by these. | provides worldwide, broad statistics, but it might not provide climate-specific content for more specialized advice. |
| AGROPOLIS: The Role of Urban Agriculture in Addressing Food Insecurity in Developing Cities | highlights the value of community participation in urban farming, | Its primary focus on developing cities may restrict its applicability to other urban settings. |
| Urban Agriculture: Sustainable Cities, Jobs, and Food | explains how urban farming might reduce food insecurity in underutilized areas. | It might not offer extremely detailed information for climate conditions. |
| Vertical Farms: From Vision to Reality | talks about cutting-edge vertical farming, which provides creative answers for constrained urban areas. | Accessibility for all users is restricted by high upfront expenses |
| Community-Based Urban Agriculture for Food Justice: A Review | demonstrates the importance of community-driven urban farming, which is consistent with the project's emphasis on resource exchange and interpersonal relationships. | highlights local-level projects that might not apply well to bigger cities. |

|  |  |  |
| --- | --- | --- |
| Resources | Benefits | Limitations |
| HTML, CSS, JavaScript | Frequently chosen, interactive, and responsive | demands cautious performance management. |
| Node.js, Express | Effective, expandable, and suitable for dynamic content | can be difficult for beginners to set up. |
| SQL | Strong and effective data management | Scaling can be challenging for large data sets. |

**Critical Analysis**

Evaluations of technology and literature present a comprehensive understanding of the project's requirements and possible solutions. The literature reviews highlight how crucial it is to use urban farming to address food security. Whereas the technology reviews offer a plan for putting the project's digital elements into practice. The chosen technologies give a reliable and scalable solution by matching the project's goals. Critical insights gathered from the reviews will direct the approach. keeping the project's goal of providing a workable and significant solution to world hunger.

# **Methodology**

**Design**

**Artefacts**

**Education Website:**

The main platform will be the website, which will give consumers instructions on how to grow and harvest crops in various climates. The design will prioritize navigation and user accessibility:

* **Responsive Design:** The responsive layout will be made using HTML, CSS, and JavaScript to adjust to various devices. Such as mobile devices, tablets and desktops
* **Content Structure:** The website will have different sections regarding climate specific growing guides, resource sharing and community forum.
* **User Interaction:** Climate-specific farming advice and interactive crop calendars are examples of dynamic features made possible by JavaScript.

**Resource Sharing Application:**

The application will make it easier for community members to trade farming equipment such as tools, seeds, and containers:

* **User Authentication**: Secure user accounts and transactions are guaranteed by the usage of Node.js and Express.
* **Item Listing**: The resources that are listed by users will be kept in a SQL database, featuring a search tool for easy access.
* **Messaging and Notifications**: With notifications for updates and new listings, built-in messaging will allow for direct user-to-user conversation.

**Community Forum:**

The community forum will be a vital tool that allows users to interact, exchange progress, and ask for guidance

* **User Interaction:** By enabling users to post queries, exchange feedback, and offer comments, the forum will promote teamwork.
* **Showcase Progress:** Users can create a space for learning and inspiration by uploading pictures and updates about their work.
* **Tools and Technologies:** SQL is used to store user information and postings, and Node.js and Express are used for backend management.

**Urban Farming Architecture Plans:**

These designs will highlight creative ways to make the most of constrained urban areas

* **Illustrative and Practical Designs:** The concepts will include rooftop gardens and modular vertical systems, along with visual aids to help users optimize the potential of urban farming.
* **Website:** The designs will be available on the website, providing users with practical advice on how to apply urban farming ideas.

**Testing and Evaluation**

The artifacts will be tested to make sure they satisfy user requirements and project objectives.

**Functional Testing:** Each artefact (website, application, community forum and architecture plans) will be tested for functionality.

* **User testing:** through surveys and interviews, target users—urban farmers and other interested parties—will offer input to pinpoint usability problems.
* **Bug Tracking and Fixing**: Any problems that occur during testing will be noted and fixed right away.

**Performance Testing**: Performance testing will be done on the website and app to make sure they can manage multiple users properly.

* **Load Testing:** To assess performance under pressure, simulate multiple users using the website
* **Database Performance**: Speed and efficiency will be prioritized in SQL database queries, particularly when dealing with big information.

**Security Testing**: Making sure there’s data protection.

* **Data Protection**: Private user information will be safely saved and encrypted.
* **Authentication**: To stop unwanted access, the login process will be examined.

**Evaluation**: The following criteria will be used to assess the project's success.

* **User Engagement**: The number of users that participate on the website.
* **Educational Impact**: Feedback on the website structure, the design, if users are interested in taking on the information.

**Project Management**

**Project Phases**

* **Phase 1**: Planning and Design - Make wireframes and prototypes and specify objectives, features, and technologies.
* **Phase 2:** Development - Construct the website and application while integrating the database, frontend, and backend.
* **Phase 3**: Testing - Test for functionality, performance, and security.
* **Phase 4:** Deployment - Launch the website and monitor for issues.
* **Phase 5**: Improvement - Get feedback and make the required adjustments.

**Technologies and Processes**

The technologies selected will ensure the project's objectives are effectively achieved. Why each technology was chosen is as follows:

**Frontend Technologies (HTML, CSS, JavaScript)**

* These technologies were chosen because to their ease of use, broad application, and ability to produce dynamic, responsive, and simple designs. JavaScript will make interactivity possible, allowing for features like a crop calendar and climate-specific advice.

**Backend Technologies (Node.js, Express)**

* Node.js allows for the effective management of numerous jobs, which is why it was chosen. Express offers a simple framework that is perfect for controlling resource sharing and user authentication.

**Database Technology (SQL)**

* For organized data, such user profiles, farming guides, and resource listings, SQL is dependable and ideal. It's ideal for project database management because of its ability to manage complex queries and data linkages.

# **Implementation**

**Development Approach**

Compared to my original intention, the process of developing the Urban Farming Solutions platform was very different. Health issues that developed during the project prevented me from following to the sprint approach that I had initially planned. I worked in phases that were flexible, which ended up being beneficial. As I worked on the design, it enabled me to adjust and enhancements.

**Project Structure and Organization**

As I was creating my project, I arranged it into folders that made sense to me. This is how it all turned out:

A screenshot of a computer

AI-generated content may be incorrect.

* **Clips** - AI-generated video guides flowlabs - google
* **Images** – Images from unsplash, these were for guides/ui
* **Guide** – HTML files for each crop to grow
* **App**.js – Frontend JavaScript
* **Climate**-api.js – weather api integration
* **Package.json** – project dependencies
* **Index**.**html** – website page
* **Other pages** – register, profile, architecture and progress-dashboard.html

Instead of making each growing guide dynamically, I made separate HTML files for each. Although it may appear outdated, this allowed users to save the sites for later use and made them load faster. something that I believed would be quite helpful for individuals who are gardening.

**Development Timeline and Phases**

To be honest, the development was chaotic at times. I would get a concept for the interface and then keep switching between frontend and backend work. I put it into practice, realized I required backend support, switched to it, and then jumped back to improve the appearance. Although it wasn't the most structured strategy, it kept me interested and made it possible for me to see results right away.

**Phase 1: First Setup and Database Selection (Weeks 1-2)**

I made a significant decision early on that had an impact on the entire project. Since MySQL was what we had studied in class, it was my initial plan. But after a few days of battling with foreign keys and table relationships, I chose to give MongoDB a try, which was recommended by a friend. I had to learn something new, so this was a little frightening, but it was much simpler for this project:

mongoose.connect('mongodb://localhost:27017/urban-farming', {

useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => {

console.log('Connected to MongoDB successfully');

}).catch((error) => {

console.error('MongoDB connection error:', error);

process.exit(1);

});

The ability to simply store data the way I imagined it was the best feature of MongoDB. I didn't have to make a ton of junction tables to link things together because users have profiles and resources have owners.

**Phase 2: Development of Core Functionalities (Weeks 3-5)**

It was around this stage that I truly got into my developing flow. but also, where I was constantly rethinking the design. After building anything and evaluating it, I would say, *"This doesn't look right,"* and then reconstruct it. Using the main homepage alone, this most likely occurred four or five times, especially for the “growing guides” as I only had 3 crops to start with and how the API looked as well as how I’d make the “Architecture Plans” look.

I wanted to ensure the security of user passwords. I therefore took the effort to study session management and crypt. And added authentication systems:

// User registration - making sure passwords are hashed

app.post('/register', async (req, res) => {

const { username, email, password } = req.body;

try {

// First, check if someone already has this username or email

const existingUser = await User.findOne({

$or: [{ username }, { email }]

});

if (existingUser) {

return res.status(400).json({

message: 'Username or email already exists'

});

}

// Hash the password - this was new to me!

const hashedPassword = await bcrypt.hash(password, 10);

// Create the new user

const newUser = new User({

username,

email,

password: hashedPassword

});

await newUser.save();

res.status(201).json({ message: 'User registered successfully' });

} catch (error) {

console.error('Registration error:', error);

res.status(500).json({ message: 'Registration failed' });

}

});

**Phase 3: Improvement of Features (Weeks 6-8)**

I had a few family members and friends test the website, and they provided me with feedback.

One person mentioned, *“This is great, but what if you add a tracking progress that allows user to add their crops, and they can track the growing cycle*.” I then understood that I required a feature that tracked my progress. Something that I haven’t originally thought of or planned. Due to my health conditions I had more time to adjust, which helped me implement this feature.

I first used local storage to construct the progress tracking because I needed it to function even when users weren't signed in.

// Original implementation - using localStorage

document.getElementById('add-crop-form').addEventListener('submit', (e) => {

e.preventDefault();

// Get all the info from the form

const cropData = {

\_id: Date.now().toString(), // Simple ID generation

cropName: document.getElementById('crop-name').value,

variety: document.getElementById('variety').value,

plantedDate: document.getElementById('planted-date').value,

location: document.getElementById('location').value,

expectedHarvestDate: document.getElementById('expected-harvest').value,

notes: document.getElementById('notes').value,

status: 'growing'

};

// Add it to our saved crops in localStorage

crops.push(cropData);

saveCropsToStorage();

// Clear the form

e.target.reset();

toggleAddForm();

// Update what the user sees

displayCrops(crops);

updateStats(crops);

});

A screenshot of a computer

AI-generated content may be incorrect.

But after further testing and consideration, I became aware of the significant drawbacks of this approach. Users were unable to view their progress across devices. and they would lose everything if they deleted their browser's data. Additionally, several testers expressed how much they would like to view their history of improvement over time. To incorporate the progress monitoring with the user authentication mechanism, I redesigned it entirely. Users now see a clear notification when they access the progress dashboard without being logged in: “Login Required”

Figure 1: progress before

A screenshot of a green and white website

AI-generated content may be incorrect.

Figure 2: progress before

The change greatly strengthened the feature. Now, users can:

* View the status of their crops on any device.
* Won’t lose data
* View their entire gardening history
* Delete/update the status of the crop

**Technical Implementation Details**

It was a journey to integrate the weather API. My initial impression of the Open-Meteo documentation was that I felt extremely lost. I already had struggles with any type of APIs and this was no joke, just figuring out what parameters I needed took hours of my time. To comprehend how to arrange the API, I watched several YouTube videos and gained assistance from AI. The realization that I needed to consider things step-by-step was the turning point:

// Getting weather data - this took me forever to figure out!

async function fetchWeatherData(latitude, longitude) {

try {

// Building the URL with all the parameters

const url = `https://api.open-meteo.com/v1/forecast?latitude=${latitude}&longitude=${longitude}&daily=temperature\_2m\_max,temperature\_2m\_min,precipitation\_sum&current\_weather=true&timezone=auto`;

const response = await fetch(url);

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const data = await response.json();

return data;

} catch (error) {

console.error('Error fetching weather data:', error);

throw error;

}

}

Making recommendations for crops based on weather data was the true challenge. I developed a straightforward approach that classifies climates and recommends suitable crops:

// Figuring out what to recommend based on the weather

function getCropRecommendations(weatherData) {

const currentTemp = weatherData.current\_weather.temperature;

const maxTemps = weatherData.daily.temperature\_2m\_max;

const minTemps = weatherData.daily.temperature\_2m\_min;

// Calculate averages - basic math but it works!

const avgMaxTemp = maxTemps.reduce((sum, temp) => sum + temp, 0) / maxTemps.length;

const avgMinTemp = minTemps.reduce((sum, temp) => sum + temp, 0) / minTemps.length;

const avgTemp = (avgMaxTemp + avgMinTemp) / 2;

// Simple logic to categorize the climate

let climateType = "";

if (avgTemp > 25) {

climateType = "hot";

} else if (avgTemp > 15) {

climateType = "warm";

} else {

climateType = "cool";

}

// Return appropriate recommendations

return recommendations[climateType];

}

**Resource Sharing - Building Trust**

The resource-sharing tool that my supervisor provided me, they gave example such as, sharing seeds and tools, that could be neighbours or anyone local, I was concerned for privacy, so I tried to balance that.

// Creating a new resource listing

app.post('/api/resources', async (req, res) => {

// Make sure they're logged in first

if (!req.session.userId) {

return res.status(401).json({ message: 'You need to log in first!' });

}

try {

// Get their username to display with the resource

const user = await User.findById(req.session.userId);

// Create the resource with their info

const resource = new Resource({

...req.body,

owner: req.session.userId,

ownerName: user.username // Only showing username, not email

});

await resource.save();

res.status(201).json(resource);

} catch (error) {

res.status(500).json({ message: 'Something went wrong!' });

}

});

**Dealing with Unexpected Challenges**

**AI Video Solution**

Not being able to produce the content I had originally intended was one of the biggest disappointments. I intended to personally plant every crop, take pictures and films to record the process, and let users know about that actual experience. But I couldn't do this because of my health problems.

This caused a lot of stress as I had no plan to help with the guides, as I originally planned to have some sort of illustrations for the crop guide. At the start I only had three crops which wouldn’t have been hard to do myself but due to my health problems I was at a loss. I spoke to a close friend of mine and explained the struggle I was going through and that I didn’t have much time to finish off my work. He mentioned I should use AI generated clips for my guide. I learned that FlowLabs could produce AI videos. Although it was different from my original plan, I was still able to offer visual aids for each crop. Despite it's not "real" footage, I believe the videos that each growing guide has that demonstrate the essential procedures nonetheless provide value. Some generated clips came out a bit weird but from testers they all said *“For what its worth it does a pretty good job, even if it’s not perfect”*

Here’s an example of a prompt used in step 2 of growing lettuce – “**Step 2: Planting Requirements”**

*"Show lettuce planting setup: Display container with drainage holes, pour potting mix into container, sprinkle lettuce seeds on soil surface, lightly cover with thin soil layer, mist gently with spray bottle. Each action 1-2 seconds, smooth transitions. Bright indoor light, clean workspace with planting supplies visible. No text or music."*

*This was based on the steps given on the guide*

*A screen shot of a video

AI-generated content may be incorrect.*

Figure 3: Guide step

**Mobile Responsiveness Frustrations**

It was more difficult than I thought to make the website responsive. On the computer, everything appeared flawless, but when I tested on my phone, things didn’t seem right, some text would overlap, search bars were small or to big. However, using the website on the phone and iPad still worked, I didn’t have issues on my end.

I tried to give it a go with styling the frame for other devices:

/\* Mobile responsive design implemented in the index file\*/

@media (max-width: 768px) {

nav ul {

flex-direction: column;

align-items: center;

}

nav ul li {

margin: 0.5rem 0;

}

.features {

grid-template-columns: 1fr;

}

.category-cards {

grid-template-columns: 1fr;

}

.crop-grid {

grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));

}

.user-menu {

position: static;

text-align: center;

margin-top: 1rem;

background-color: transparent;

}

}

I tried to make a separate file named “mobile-style.css” but due to time I didn’t get to finish the work, if I put more time into it, I could have finished the styling and made it better for other devices. My health did put me down a lot and I sometimes would leave things to last minute of take a break from doing anything due to not feeling motivated.

**User Testing Adventures**

1. **Register** - To see what would happen, my cousin attempted to register for two accounts using the same username.The site gave my cousin a text saying that “Username or email already exists”. Which was a good start for validation
2. **Personal data –** another tester used the progress tracker, as mentioned it was first public, so when the tester added crops to their progress tracker and refreshed the page, they lost their progress. In which later I added all progress to a local storage which was linked to the user’s profile.
3. **Password –** A key function I missed out on was how secure a password would be, another tester registered and made a password that was “123”, even I was shocked to see the result in which informed the tester that I would change this immediately. Which I’ve changed in my “register.html” file.
4. **Weather api -** There would be no results when someone searched for a city.no mistakes, no outcomes. I debugged this for hours, adding console logs everywhere. I then realised that the function name I was calling was incorrect. For example:

// What I had (wrong):

getClimateRecommendation();

// What it should have been:

getClimateRecommendations(); // with an 's'!

once I got the api working, the same tester would see that there was an issue when pressing “enter” on their keyboard and then clicking on “get recommendation” there would be two different outputs in which made the site crash and to fix it, the tester would have to refresh the page. Due to time, I couldn’t solve the issue. For the future I’d come back to it and solve the problem,

**A screenshot of a login form

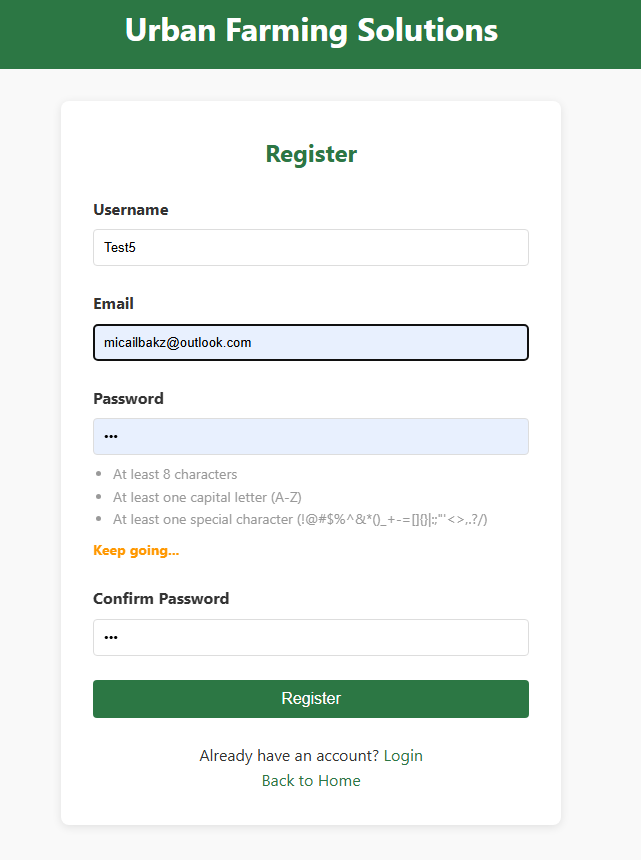
AI-generated content may be incorrect.**

Figure 5: register after

Figure 4: register before

**Database Design Decisions**

For this project, using MongoDB proved far more user-friendly than using SQL. I could organize data in the manner that came to mind naturally:

// User schema - keeping it simple

const userSchema = new mongoose.Schema({

username: { type: String, required: true, unique: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

bio: String, // Added later when users asked for profiles

createdAt: { type: Date, default: Date.now }

});

// Resource schema - everything about a shared item

const resourceSchema = new mongoose.Schema({

title: String,

category: String,

description: String,

quantity: Number,

location: String,

contactInfo: String,

owner: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

ownerName: String // Denormalized for easy display

});

**Tools and Workflow**

I used my tools in a straightforward manner:

* I did all my coding in Visual Studio
* Chrome DevTools, this was mainly for troubleshooting, I spent a great deal of time in the console tab, using “f12” on my keyboard to have access, this showed me more detail if my JavaScript were running correctly, making sure my divs were updating correctly and identifying any issues. Which I had plenty with the weather API.
* Although I am aware that I should utilize appropriate project management tools, I find that handwritten lists are effective, not be motivated to get on my computer which was due to my health problems.
* GitHub will be added soon with all my work as well a project task tracking.

To demonstrate the development process in a more professional manner, a genuine Kanban board will be created on GitHub. However, I'll admit that I just utilized sticky notes on my wall during development. This helped as I would see them whenever I’m in my room, this gave the boost I needed to work on my project.

**Reflections on the Implementation Process**

Upon reflection, this implementation was somewhat rough. In truth, the phased plan that developed out of necessity (because of my health) was more effective than strict sprints might have been. I was able to employ my creativity, react to user input, and change course, when necessary, as seen by the AI clips, progress tracking for crops.

What is the most important lesson? features that testers/users requested, such as progress tracking

Would I act differently? Of course. I would begin with mobile design rather than desktop. I would set up GitHub right away. Instead of wasting time on MySQL, I would probably continue using MongoDB from the beginning. Overall, though, I'm proud of what I created because it's functional, usable, and may even inspire someone to cultivate their own food, which is cool as I’ve come from an engineering side. So, to think I came this far, I’m proud.

# **Evaluation and Results**

**Introduction**

**Getting Started with Testing**

When the time came for me to test all I had built, I became aware that my flexible approach, which was dictated by my health, had required me to test constantly as I developed. Since I was unable to predict when I would have productive workdays, As I was building it, I had developed the practice of testing every little component and, if I could, getting feedback. This testing method, which was developed out of need rather than design, really identified several problems early.

**How I Tested Everything**

Checking that features functioned, I went through every section of the program as if I were an ordinary user, attempting to break things (Sometimes succeeding).

**Login System:**

* Verified that passwords were, in fact, being hashed (luckily, they were).
* When someone attempted to create two accounts using the same login, the system detected it.
* When incorrect passwords were used, the error messages made sense.

At first there was an issue that I wasn’t aware of which regarded when registering – passwords. The system worked perfectly well with a password like "123" when you signed up. I totally missed this easy issue when testing the registration code—probably because I was under a lot of stress at the time. When a friend made an account using that exact password, she brought it to my attention. so, I quickly added some basic requirements.

The password checker looks for:

* Eight characters is the minimum, but there is no upper limit.
* One capital letter (A-Z) at minimum
* One special character (!@#$%^&\*()\_+-=[]{}|:;"'<>,.?/) at minimum

**Growing Guide:**

* Verified that all the information appeared as intended.
* Verified that the AI videos played
* Tested switching between the guides.
* Things were a little disorganized once I looked at everything on my phone, but still worked only tiny flaws

Because each instruction was created as a separate HTML file, they loaded incredibly quickly. However, I had to change fifteen files after discovering a typo in the footer. It wasn't the best choice I made.

**Weather API**

* Verified that the weather service could be reached.
* Checked my temperature average calculation, which I had done incorrectly the first time.
* Confirmed that it recommended the appropriate crops for various climes. (still needs a bit work for future plans)

When users and I searched for the weather. Nothing came up, just confusion, loading spinner, no error. It made a big impact to add appropriate error messages. The API is still unfinished. It works but is still very buggy. I found out the reason behind the issue. The issue was that two systems running where the user or I would press “Enter” on the keyboard it calls getClimateRecommendations() from the “climate-api.js” and that would work fine. However, when the user or I would click on “get recommendations” it wouldn’t work. I realised that there were two ways of displaying the results. Due to time, I didn’t get to solve this issue. Again, for the future I’ll give this method more time and solve it next time to have a smooth experience.

**Resource Sharing App**

* Added, viewed, edited, and deleted resources were tested.
* Ensured that resources appeared under the appropriate user.
* Verified again that emails remained private (only usernames were displayed).
* Ensured that once resource was deleted it was gone

**How Quickly Did Everything Go?**

I was unable to conduct thorough performance testing without a suitable server configuration. However, I took what I could:

**Loading Speeds**

* Homepage – up to 2 seconds
* Growing guide page – up to a second
* Resource page – would vary sometimes it be quick or a bit slow

Throughout the loading process, the speed was good, didn’t have issues even with testers.

**Database Performance**

* Found it easy to search upusers

Leaving MySQL was a wise decision. No more complex searches that combine five tables.

**Security Checks**

* Made ensuring that passwords were not stored in plain text in the database.

Figure 6: encrypted password

* Verified that you were indeed logged out after logging out.
* Made sure that the email addresses of other people were hidden.
* Confirmed that some pages required you to be logged in.
* Verified that people couldn't view each other's information.

**Real People Using My Site**

**Who tested it:**

* Three family members, two sisters and one Brother, could have been better as one of sisters tested with me and helped with tips and any issues. My other two siblings made test accounts, with usernames of “Test1” and “Test2” more accounts were made but also deleted to test if deleting accounts worked.
* 5 friends tested, but they wanted to be hidden, one account still is active while the other one was deleted.
* This would be a whole lot better if I had someone who gardens and get accurate feedback.

**What I Asked Them to Do**

I assigned the same tasks to everyone and monitored the results.

* Create an account.
* Locate the guide to producing tomatoes.
* Obtain local weather forecasts
* Add something to share with other people.
* Add progress (play around with it, keep/delete it)

**What They Loved**

* They all mentioned that it was easy to use
* They mentioned that the website looked good, one said that I should add images of flowers or plants on the side to match the theme of growing crops.
* AI clips – they found it funny but gave positive feedback after finding out that the clips were free to generate. (had to make multiple accounts to get a free trial)
* They mentioned that the website was fast.

**What Work Was Needed**

* When using a mobile phone, the website to some looks cramped or text not being aligned correctly, so work on mobile styling needs to be looked at in the future.
* Progress tracking could have another feature which I realised and that could be having a leaderboard to make it more engaging for users, this could help push users to grow more.
* The most important work that needs to be done is the weather API. I managed to get the API to work. However, a lot needs to be done to make sure that the API is smooth. Improve the recommended crops based on location of the user. Give better tips for growing.
* Some didn’t like that their number was out there, so maybe have a feature where users can message each other through the app, instead of giving out personal info.

**The Numbers**

**Success rate:**

* 100% of people created an account
* Crop guidance finding was very simple
* Weather API was working around 75%
* Listing an item was simple
* Finding resources was simple as users can see when going on resource page

**How much time It took:**

* Users took around a minute to sign up
* Users finding crop guides didn’t take much time around 5-10 seconds
* Checking weather climates varied as some tested how it worked, that being different cities (was limited)
* Listing an item or crop progress didn’t take long either around 2 minutes

**Reviewing My Code**

**My Strong Points**

* Organization – the backend and frontend were well separated
* Naming - names for my variables that were descriptive.
* Error Handling – these were handled very well throughout testing
* Security - Passwords were secure, on both frontend and backend, could add more as mentioned before with resource sharing – instead of phone numbers, send messages to other users.

**Why Was It Messy?**

* Copy pate of code - There was a ton of repetitive code in the growing guides.
* Weather API – two systems would run meaning there would be two results depending on how the user use the API, e.g. if the user presses “Enter” on their keyboard or click on “get recommendation”.
* Resource sharing app – could be better as mentioned above, users should be able to send messages to other users when see an item listed instead of personal info being public as some users might not want their contact number out there.

**Did It Actually Help Anyone?**

**Learning Impact:**

* “I had no idea tomatoes required so much room”
* “I can understand why my herbs died years ago based on your API thingy”
* “The ai clips were funny but I can get an understanding of what to do based on the guide and the clips”
* “When I’m out, I often come across architectural plans, so your explanation—complete with images and reasoning—was especially helpful.”

**Action taken:**

* Unfortunately, only my sisters said they’d use this; other testers were open to use this website if they had the “energy” to grow their crops.

**In contrast to my initial plans, how did I do?**

**What I did:**

* Learning Website – created 12 growing guides with clips supporting the guide.
* Resource sharing app – users can listen items, delete their listed item on their profile.
* User accounts – accounts are made and secure – login and logout work
* Tracking Progress – users can list their crops on their tracking progress, they can update the status of their crops, “growing”, “harvested” and “failed”.

Half did/could do differently:

* Weather API – works but could be better if I gave it more time to make it smooth
* Mobile styling – could have added the styling into my code from the beginning and updated each html file

What I missed out on:

* Community forum – lack of time and after looking back at it, this wouldn’t have been a good approach.
* Server – make the website work offline instead of localhost.

**Results**

**A screenshot of a website

AI-generated content may be incorrect.Main Page:**

Figure 7: First part of main page

**A screenshot of a web page

AI-generated content may be incorrect.**

Figure 8: second part of main page

**A screenshot of a login form

AI-generated content may be incorrect.Login/Register Page:**

Figure 9: Login Page

**A screenshot of a login screen

AI-generated content may be incorrect.**

Figure 10: Login attempt Page

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.**

Figure 11: Register Page

Figure 12: Register attempt Page

**A screenshot of a food

AI-generated content may be incorrect.Growing Guide Page:**

**A screenshot of a phone

AI-generated content may be incorrect.**

Figure 13: Growing guides

Figure 14: Vegetable guide

**A screen shot of a video

AI-generated content may be incorrect.**

Figure 15: tomatoes guide step 1

**A screenshot of a video

AI-generated content may be incorrect.**

Figure 16: tomatoes guide step 2

**A screenshot of a video

AI-generated content may be incorrect.**

Figure 17: tomatoes guide step 3

**A screenshot of a video

AI-generated content may be incorrect.**

Figure 18: tomatoes guide step 4

**A hand planting a plant

AI-generated content may be incorrect.**

Figure 19: tomatoes guide step 5

**A hand holding a tomato

AI-generated content may be incorrect.**

Figure 20: tomatoes guide step 6

**A screenshot of a green and white website

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

Figure 21: weather API

Figure 22: Example of London

**A blue background with black text

AI-generated content may be incorrect.Resource Sharing App Page:**

Figure 23: Resource sharing

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 24: Resource sharing Page

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 25: sharing a resource

**A screenshot of a website

AI-generated content may be incorrect.Urban Architecture Plans Page:**

**A screenshot of a website

AI-generated content may be incorrect.**

Figure 26: Urban Farming page

Figure 27: vertical farming

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 28: window farming

**A screenshot of a website

AI-generated content may be incorrect.**

Figure 29: window farming

**Profile page:**

**A screenshot of a green and white website

AI-generated content may be incorrect.**

Figure 30: profile page

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 31: user can delete their account

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 32: user can edit their profile

**Finalizing the evaluation**

Urban Farming Solutions accomplishes its goals. Educate people about urban food production. Some components are just put together, and not all of it is completely constructed. However, it functions. It was utilized, knowledge was gained, and some even mentioned they’d start growing.

The testing demonstrated that it was wise to keep things quick and easy. The choice of MongoDB was excellent, and the weather integration truly adds value. and sharing resources with neighbours is something that people genuinely desire.

However, it also highlighted my mistakes. The focus should have been on mobile, progress tracking must be completely redesigned, as well as the weather API.

Despite all the flaws, users rated the website very highly. That indicates to me that the main idea is good: people are looking for easily available information about urban gardening. All they require is an improved mobile experience and a means of communication.

This platform demonstrates that helping people grow food doesn't require advanced sensors or artificial intelligence. At times basic instructions, local weather reports, and a means of sharing seeds with your neighbour is sufficient to initiate an urban farming endeavour.

# **Conclusion**

Reflecting on the Urban Farming Solutions project, what a ride it has been. An ambitious attempt to establish a complete urban agricultural platform turned into a lesson in flexibility, software development reality and user-cantered design.

The development of an educational platform that supports urban farming as a means of addressing food insecurity was the project's primary goal, and it was accomplished. The finished system has a weather-based recommendation system, 12 expanding guides (more could have been added but would have taken a lot of time) with AI-generated videos, safe user verification, and community resource sharing. It received 3.5 out of 5 from 8 testers, and more significantly, in which two of the testers said they’d start growing.

Naturally, not all my original ideas were included. For starters community forum was never added. Before I’ve addressed that community forum wasn’t needed. Looking back with the feedback of users sending messages to each other regarding listing items, maybe and forum could be useful as users can help each other and possibly add more guidance regarding gardening or even what approaches they went around harvesting a crop. Mobile use as well as other devices but it isn’t at its finest. Still gets the work done. Same with the weather API there are some limitations but it all still works take it make user friendly which was a goal of mine from the start.

The actual development process was affected by unforeseen obstacles. I had to go from a strict sprint methodology to a more flexible one due to health concerns. On some days, I would work on intricate backend logic, and on others, I would oversee small CSS adjustments. I learned from this flexibility, which came from necessity, that being able to keep going forward is more essential than having perfect plan.

There were some significant turns in the technological journey. Switching midway through a project from MySQL to MongoDB seemed like a failure. During the beginning of the project, I did lose all hope. However, it made things simpler and more in line with my data design. The location-based growing advice offered by the weather API, that almost broke me are now helpful. (somewhat helps – still needs improving)

**Future Work:**

Add a leaderboard for the progress tracking, allows a more engaging environment for users to be involved in. could possibly add badges for who’s got the best rate, the most crops harvested etc. This could make user experience more fun.

In addition to mobile/other devices is equally urgent as all users deserve to have the same experience, the full use of the functionality. This entails redesigning the entire touch interface, adding responsive CSS alone is not enough.

Add community forum, this will users to share photos, ask other users questions, have a topic to talk about regarding gardening. By doing this, the platform would change from a source of knowledge to a community. Location-based search, ratings, and messaging might be added to the resource sharing system.

**Reflection**

I was pushed well beyond my comfort zone by this project. After completing tutorials, I designed a full-stack application. This development is best demonstrated by the MongoDB switch, which can identify when a well-known tool isn't appropriate and has the courage to pick up new skills in the middle of a project.

My patience was put to the test by the weather API integration. Success eventually resulted from breaking it down into smaller issues after a confusing month. Which showed that if i put more effort and time I could had the API working perfectly.

Health issues enforced adaptability Although I wouldn't have chosen, I ended up benefiting from it. This allowed me to work on my project for much longer compared to most peers in my class. Which gave me the advantage which I’m truly grateful for and can’t thank enough for my supervisor and other lecturers to understand my situation and be patient with me. I myself, is not proud of how this project came to an end as I could have done better and I don’t want to shift all the blame on my health. Therefore, this allowed me to add more features to my project based on feedback on testers.

My perspective on software development has been changed due to this project. It has nothing to do with advanced frameworks or coding. It's about assisting others in solving real-world problems. Unstable mobile design, weather API not being perfect, and missing forums are some of Urban Farming Solutions' shortcomings. However, it is real, functional, and aids in food production. Every annoying issue, every difficulty with the API, and every late night was worthwhile.

I ended up creating something more practical than I had anticipated. It demonstrates that imperfect solutions are preferable to perfect ones that are anticipated. If even a single individual benefits from our project by being able to grow their own food and feel safer about it, I'll consider it a total success at that point.

# **References**

**[1] HTML/CSS Structure and StylingBasic HTML/CSS Layout**

**MDN Web Docs:** <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout>

**YouTube: "HTML & CSS Full Course" by freeCodeCamp** – <https://www.youtube.com/watch?v=mU6anWqZJcc>

**[2] Responsive Design**

W3Schools: <https://www.w3schools.com/css/css_rwd_intro.asp>

YouTube: "Build a Responsive Website with HTML & CSS" by Kevin Powell –<https://www.youtube.com/watch?v=p0bGHP-PXD4>

**[3] CSS Grid Layout**

MDN Web Docs: <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout>

YouTube: "CSS Grid Layout Crash Course" by Traversy Media - <https://www.youtube.com/watch?v=jV8B24rSN5o>

**[4] Express.js Server Configuration Setting up Express.js Server**

Express.js Documentation: <https://expressjs.com/en/starter/hello-world.html>

YouTube: "Express JS Crash Course" by Traversy Media - <https://www.youtube.com/watch?v=L72fhGm1tfE>

**[5] Serving Static Files**

Express.js Documentation: <https://expressjs.com/en/starter/static-files.html>

YouTube: "Express.js Tutorial: Serving Static Files" - <https://www.youtube.com/watch?v=SccSCuHhOw0>

**[6]** MongoDB Integration/MongoDB with Express

MongoDB Documentation: <https://www.mongodb.com/docs/drivers/node/current/>

YouTube: "MongoDB with Node.js Crash Course" by Traversy Media - <https://www.youtube.com/watch?v=-56x56UppqQ>

**[7] Mongoose ODM**

Mongoose Documentation: <https://mongoosejs.com/docs/>

YouTube: "MongoDB & Mongoose with Express" - <https://www.youtube.com/watch?v=bhiEJW5poHU>

**[8] User Authentication/Express.js Authentication**

Express.js Session: <https://www.npmjs.com/package/express-session>

YouTube: "Node.js With Passport Authentication" by Traversy Media - <https://www.youtube.com/watch?v=6FOq4cUdH8k>

**[9] Password Hashing with bcrypt**

bcrypt Documentation: <https://www.npmjs.com/package/bcrypt>

YouTube: "Password Hashing with bcrypt" - <https://www.youtube.com/watch?v=O6cmuiTBZVs>

**[10] API Integration (Open-Meteo)/Fetch API**

MDN Web Docs: <https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API>

YouTube: "Fetch API Introduction" by Traversy Media - <https://www.youtube.com/watch?v=Oive66jrwBs>

**[11] Weather API Integration**

Open-Meteo API Documentation: <https://open-meteo.com/en/docs>

YouTube: "How to Create a Weather App with API" - <https://www.youtube.com/watch?v=MIYQR->Ybrn4

**[12] Resource Sharing Application/RESTful API Design**

MDN Web Docs: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

YouTube: "RESTful API With Node.js, Express & MongoDB" by Traversy Media - <https://www.youtube.com/watch?v=vjf774RKrLc>

**[13] CRUD Operations**

MongoDB CRUD Operations: <https://www.mongodb.com/docs/manual/crud/>

YouTube: "CRUD REST API using Node, Express & MongoDB" - <https://www.youtube.com/watch?v=CyTWPr_WwdI>

**[14] Client-Side JavaScript/DOM Manipulation**

MDN Web Docs: <https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model>

YouTube: "JavaScript DOM Manipulation" by Traversy Media - <https://www.youtube.com/watch?v=0ik6X4DJKCc>

**[15] Async/Await**

MDN Web Docs: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function>

YouTube: "JavaScript Async Await" by Traversy Media - <https://www.youtube.com/watch?v=PoRJizFvM7s>

**[16] Project Structure and Architecture/MVC Pattern**

MDN Web Docs: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

YouTube: "MVC Design Pattern" by Programming with Mosh - <https://www.youtube.com/watch?v=DUg2SWWK18I>

**[17] Node.js Project Structure**

Express Application Generator: <https://expressjs.com/en/starter/generator.html>YouTube: "Node.js / Express Project Structure" - <https://www.youtube.com/watch?v=oNlMrpnUSFE>

**[18] Blueberries guide**

YouTube: “How to grow your own blueberries | The RHS**“** [**https://www.youtube.com/watch?v=KOGzgCDnMmw**](https://www.youtube.com/watch?v=KOGzgCDnMmw)

**[19] herbs guide**

YouTube: “Indoor Herb Gardens” - The Definitive Guide For Beginners” by The Ripe Tomato Farms = [**https://www.youtube.com/watch?v=uHLChW4R5vw**](https://www.youtube.com/watch?v=uHLChW4R5vw)

**[20] AI videos**

Google web**:** [**https://labs.google/fx/tools/flow**](https://labs.google/fx/tools/flow)

**[21] Vertical Farming**

Google web**:** [**https://www.edengreen.com/blog-collection/how-hydroponics-can-save-our-food-system**](https://www.edengreen.com/blog-collection/how-hydroponics-can-save-our-food-system)

**[22] Window Farming**

Google web: <https://gardenculturemagazine.com/window-farming/>

**[23] Rooftop Farming**

Google web: <https://www.urbanvine.co/blog/5-reasons-why-urban-farming-just-works-on-rooftops>

# **Appendices**

Appendices appear after references. Your appendices depend on the nature of your project. **Do not assume people will read your appendices.** Even if you direct them to do so in your main text, appendices are considered additional information and should not be relied upon to understand your main body of work. Refer readers to an appendix using a phrase such as *see Appendix A for further details*.

The following documents **must** be included as references:

* Your Project Proposal.
* Evidence of your use of a project management tool.
* A description of how to access any technical output. **It is strongly recommended you use GitHub or something similar to do this.**

Any important communications between you and external stakeholders -- **please ensure private data is removed and communications anonymised.**