UNIVERSITY OF CAPE TOWN

MINOR DISSERTATION

Blockchain-enabled weather index crop insurance for smallholder farmers

Authors:
Tatenda A. Muvhu

Supervisor:
Julian Kanjere

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Philosophy in Financial Technology (FinTech)

August 5, 2024

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy to the University of Cape Town. It has not before been submitted for any degree or examination.

Signed by candidate

August 5, 2024

Abstract

Weather index crop insurance is an innovative approach to protect farmers against financial losses incurred due to adverse weather events that reduce crop yield. This type of insurance has the potential to be more affordable to underprivileged communities in emerging economies as it incurs lower transaction costs compared to conventional insurance, and it reduces both adverse selection and moral hazard in insurance provision. This thesis designs and implements a weather index crop insurance platform on the Algorand blockchain to enable low-cost, secure, transparent, efficient and easily verifiable transactions between weather data providers, insurers and farmers. To guide the implementation, an examination of the present state of weather index crop insurance in South Africa, its benefits and limitations, and the possibility of using blockchain technology to tackle some of the challenges faced by smallholder farmers when using conventional crop insurance methods is done. These challenges include long claim cycles, costly claim disputes and paper-heavy policies. The prototype weather index crop insurance platform improves the efficiency of the insurance process for weather data management, insurance policy management and insurance payouts processing. The platform stores transactions on the blockchain for transparency and verifiability and streamlines the insurance payout process with the use of auto-triggered payouts when weather data-dependent strike conditions attached to an insurance policy are met. Additionally, it digitizes policy management, eliminating the need for paper. Tests on the platform show that a farmer's request to join an insurance policy is finalized in an average of 9.13 seconds, from clicking "Join" to confirming the transaction and logging details on the blockchain. This process incurs a fixed transaction fee of 0.001 ALGO (approximately ZAR 0.002 as at 06/08/2023). Compared to traditional insurance applications, which take days to confirm, this is significantly faster and cheaper. Payouts on the platform are processed with an average processing time of 16 seconds from the detection of a strike event, in sharp contrast with conventional insurance where payout processing can take days. Further, in contrast to traditional insurance schemes, the weather index crop insurance platform requires fewer communication rounds between insurance actors. This benefits farmers with a simplified insurance policy enrollment process, reduced administrative burden, and faster payout processing. Load tests show that deploying the platform behind a load balancer, which distributes workload across multiple resources, enhances response time and overall performance, enabling it to handle more concurrent requests.

Keywords: Agricultural Risk Management; Blockchain; Climate Change; Resilience; Weather Index Crop Insurance

Acknowledgements

- The Lord My Father, how can I forget the trials and tribulations you have conquered for me? I thank you for the gift of life and all other gifts that you have provided in due time and all those on the way.
- To my supervisor, Julian Kanjere, thank you for your guidance. My word of praise and forbearance is due to you.
- To my family and friends, thank you for your support.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

API	Application Programming Interface
CBDC	Central Bank Digital Currency
FSCA	Financial Sector Conduct Authority
GIIF	Global Index Insurance Facility
GND	Ground
GPIO	General-Purpose Input/Output
IFAD	International Fund for Agricultural Development
IoT	Internet of Things
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
POPIA	Protection of Personal Information Act
PPoS	Pure Proof of Stake
PWM	Pulse-Width Modulation
SAIA	South African Insurance Association
SDK	Software Development Kit
SMS	Short Message Service
SPI	Serial Peripheral Interface
SoC	System on Chip
UART	Universal Asynchronous Receiver-Transmitter
USSD	Unstructured Supplementary Service Data
VCC	Voltage Common Collector

1 Introduction

Weather index crop insurance is a parametric type of insurance that shields farmers from financial losses caused by adverse weather conditions that affect crop production and reduce yield (Collier et al., 2009). This type of insurance relies on an index of weather conditions such as temperature, precipitation, and humidity to determine the probability of crop damage and yield reduction, which in turn determines the payout to be issued by the insurer. When taking up weather index crop insurance, farmers purchase an insurance policy that pays out if a predefined weather event such as drought or excessive rainfall happens during the growing season and results in a significant drop in crop yield. In this insurance scheme, payouts are not dependent on individual losses incurred by the policyholders but on an index recorded regionally which is used as a proxy for the loss incurred in a particular region (Hochrainer-Stigler et al., 2014). The indices are formulated using independent measures, for example, the amount of rainfall or temperature (Ghosh et al., 2020). Index-based crop insurance hinges on objectively determined data which in turn eliminates the need for in-field crop loss assessments. This reduces the costs of administration of insurance claims, as the existence of weather data and set insurance policy conditions triggers payouts automatically, without the need for insurance agents having to physically verify the extent of crop loss.

In this research, I explore the potential of combining weather index crop insurance and blockchain technology to develop an insurance platform suitable for smallholder farmers, that seeks to address some of the challenges they face with traditional insurance products. These challenges include long claim cycles, costly claim disputes and paper-heavy policies. The platform is designed to facilitate secure and efficient transactions stored on the blockchain for transparency and verifiability, eliminate lengthy claim processing with instant auto-triggered payouts, and maintain the privacy and security of sensitive data through user authentication. In addition, the platform reduces dependence on paper-heavy insurance policies through the digitization of insurance policy management.

Small-scale farmers in South Africa, like their counterparts in other parts of Africa and other emerging markets, encounter significant fluctuations in crop production due to weather-related challenges like drought and floods (Mapfumo, 2007). Their plight is exacerbated by their limited access to high-yielding, disease-resistant seed varieties and essential resources like fertilizer. In order to obtain loans for acquiring agricultural inputs, these farmers would typically be required to offer their assets as collateral to banks or microfinance institutions. However, the majority of these farmers lack the necessary assets to serve as collateral. Their inability to provide collateral, combined with their heavy reliance on rainfall for irrigation, renders small-scale farmers as high-risk borrowers, leading to limited availability of loans from most banks. Their inability to afford insurance also leaves them vulnerable to the effects of climate change and adverse weather conditions.

1.1 Significance of Project

A blockchain-enabled weather index crop insurance platform has significant potential to transform the way smallholder farmers manage risk and protect their livelihoods. By leveraging blockchain technology, the platform can offer more transparent, cheaper, efficient, and accessible insurance products that are tailored to the needs of smallholder farmers in emerging economies. This can help to address the key challenges faced by these farmers, such as those arising from climate change, market volatility, and lack of access to affordable finance (Masipa, 2017).

The significance of this project lies in its potential to contribute to the creation of a more inclusive and sustainable agricultural sector in South Africa and beyond, where smallholder farmers can confidently carry out their farming operations, knowing that they have a safety net in case of unforeseen adverse weather events that lead to yield loss.

1.2 Research Question

In this research, I explore the following question:

Can the feasibility and usefulness of a blockchain-enabled weather index crop insurance platform tailored for smallholder farmers, be demonstrated through the development and evaluation of a prototype platform for weather data management, insurance policy management and insurance payout processing?

1.2.1 Research Objectives

The research objectives to address the research question, include:

- Developing a weather index crop insurance platform that leverages blockchain technology to provide a more efficient and transparent weather index insurance solution for smallholder farmers, weather data providers and insurers in emerging economies.
- Building a prototype of the system that demonstrates:
  - Use of in-field sensors to collect weather data and publish it to the blockchain.
  - Digitise insurance policy management on-chain.
  - Automate insurance payouts via the blockchain, based on predetermined weather data strike conditions attached to insurance policies, to reduce the need for time-consuming manual claims verification and processing.
- Contributing to the growing body of literature on the use of blockchain technology in the agriculture sector, and providing insights into the potential for blockchain-based solutions to address the challenges faced by smallholder farmers in emerging economies with regard to accessing insurance products.

2 Literature Review

Weather index insurance is a parametric type of insurance that pays out when certain adverse weather conditions that affect crop yield occur (Cesarini et al., 2021). Typically, these conditions are measured using weather indices based on parameters such as rainfall levels and temperature. This type of insurance can be useful for farmers, businesses, and other individuals who rely on weather conditions for their livelihoods, as it can help protect them from financial losses caused by unexpected weather events (Collier et al., 2009). As defined by National Geographic [1], weather refers to the atmospheric state at a particular location and time, with parameters such as humidity, temperature, wind speed, precipitation and atmospheric pressure being measured. Of these parameters, rainfall level and temperature are the most commonly used for weather index insurance (Afriyie-Kraft et al., 2020). The weather data for these parameters can be measured on a daily or hourly basis and averaged over a period to construct an index. Once a weather index falls below or rises above a certain trigger level set in an index insurance contract, a claim event occurs. All farmers who are insured within the same area will then receive a payout, regardless of whether they have suffered a loss or not (Hochrainer-Stigler et al., 2014).

Index insurance has gained popularity as a tool for sustainable development through risk management in recent years (Cesarini et al., 2021). This type of insurance has the potential to address some of the key challenges associated with conventional insurance schemes namely moral hazard and adverse selection (Miranda and Farrin, 2012). Adverse selection occurs when only farmers who are more likely to suffer losses purchase insurance, while moral hazard arises when insured farmers reduce their production efforts in the hope of receiving insurance payouts (Robles et al., 2021). A risk pool that is more likely to experience losses is created if only those farmers that are most vulnerable to suffer losses buy insurance, while on the other hand, if insured farmers cut back on production efforts with the aim of getting insurance payouts, insurers incur more losses. With conventional insurance, these factors can be influenced by a single person but index insurance can overcome these challenges since the indices used to determine payouts are based on factors that cannot be manipulated by a single farmer (Tadesse et al., 2015). Using variables such as temperature that can be measured remotely, opens up the possibility of crop yield impact assessments without necessarily increasing assessment costs. Using Internet of Things (IoT) sensors that can automatically collect weather data coupled with storing the data to a blockchain, a distributed ledger technology that allows for secure and transparent transactions, can greatly enhance the accuracy, transparency and verifiability of insurance claims. Miranda and Farrin (2012) in their paper on index insurance for developing countries, indicate that index insurance is perceived to have significant potential for reaching smallholder farmers in emerging

[1] https://education.nationalgeographic.org/resource/resource-library-weather/

economies for 3 primary reasons:

(i) Reduced administrative costs - compensation is determined solely based on objectively determined data, eliminating the need for in-field crop loss assessments, resulting in a significant reduction of costs associated with verifying losses.
(ii) Quick payment disbursement - with the availability of automated weather stations that can transmit weather data online, the index can be computed and reported with high frequency, such as daily, enabling immediate determination of compensations.
(iii) Minimisation of asymmetric information problems - the probability of receiving insurance payouts is not influenced by the risk profile of the farmer purchasing insurance, but solely by the likelihood that the index will trigger payouts.

A major concern with index insurance is basis risk, which is the difference between a farmer's actual losses against what an index estimates as the loss incurred (Lichtenberg and Iglesias, 2022). Index insurance can fail in cases where some farmers might receive payouts without actually incurring any losses and other farmers may not receive payouts but would have incurred losses. Sources of basis risk include poorly designed insurance products and geographical elements, such as the distance between the measurement point and the field of production (Lichtenberg and Iglesias, 2022). Basis risk cannot be eliminated from index insurance schemes since indices are constructed using average conditions in an area that will not necessarily capture individual farm conditions. Basis risk is the reason why demand for index-based insurance products has remained low. A study on index insurance piloted in Malawi and India showed participation rates between twenty to thirty percent despite substantial premium subsidies to an excess of sixty percent (Gine, 2009). The challenge for research moving forward is developing accurate indices that can predict individualised losses and incorporating failsafe mechanisms so that farmers who may not receive a payout but would have incurred losses will still be protected.

One possible solution to the challenges posed by basis risk is to use IoT sensors in individual fields to collect weather data that can then be aggregated to construct an index. Increasing the density of measuring instruments and aggregating weather data from the sensors in individual fields presents a promising solution for constructing more accurate indices in weather index crop insurance. The use of accurate indices can help to reduce the basis risk associated with conventional index insurance, which is a major contributing factor to the low uptake by farmers. By improving the accuracy of indices, farmers will have more confidence in the insurance products and be more likely to purchase them. This approach stands in contrast to traditional insurance schemes, which frequently grapple with high administrative costs, low coverage rates, and challenges related to assessing and managing risks, including losses such as self-inflicted losses caused by moral hazard.

2.1 Weather Index Crop Insurance Product Design and Payout Structure

The process of designing weather-based index insurance products can prove to be a daunting task, partly due to the fact that a lot of effort is required to identify indices that minimise basis risk by consequently being correlated to the incurred yield loss (Lichtenberg and Iglesias, 2022). To accurately model an insurance contract, one would need to start by defining an index that accurately reflects the incurred losses. For this reason, the design of weather index insurance products will vary with the indexed variable, operating conditions and the objective of the cover.

The payout structure can be structured in such a way that all the insured parties in the same area are offered the same contract terms for each unit of insurance coverage, paying the same premium rate and receiving the same payout rate with the total payout depending on the value of the purchased insurance cover. According to the International Fund for Agricultural Development (IFAD), an incremental approach can also be applied to the payout structure where the damage severity varies with the increase in deviation from the trigger (Burke et al., 2010). For example, Figure 1 below shows payout varying inversely with the amount of rainfall. As the amount of rainfall increases, the payout amount decreases from a set limit.

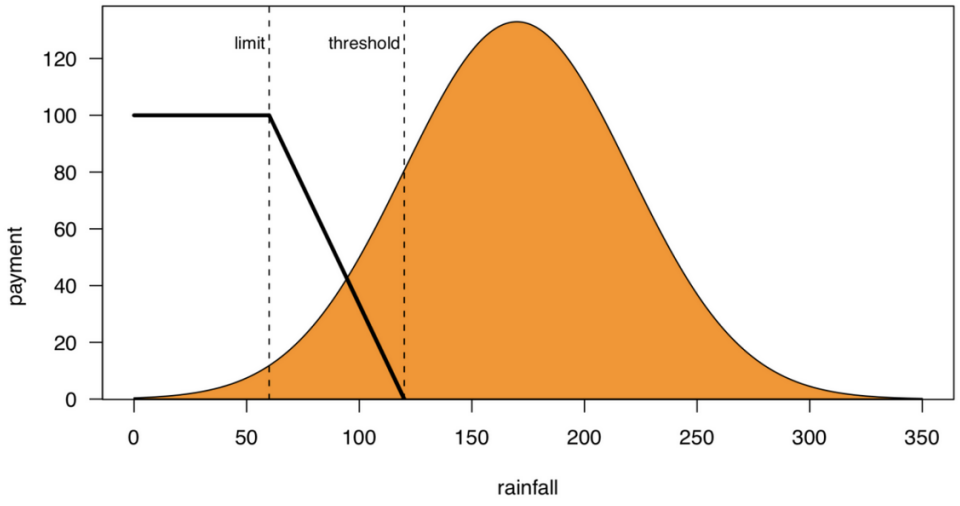

Figure 1: Index insurance payout under different rainfall scenarios, showing rainfall distribution and corresponding payout represented with the black line. As the amount of rainfall increases the payout amount decreases from a set limit up until the rainfall threshold level eligible for an insurance payout. Source: (Burke et al., 2010).

2.2 Comparing the Cost Structure of Index-based Insurance and Traditional Insurance Products

Traditional insurance provision is associated with the following costs as indicated by Burke et al. (2010):

- Pure risk - the cost of expected incurred losses to be covered by the insurance policy.
- Adverse selection - the availability of insurance generally attracts riskier farmers.
- Moral hazard - insurance gives farmers an incentive to work less and hence there is a need to monitor the yield production techniques used by insurance holders to avoid having to payout claims for self-inflicted losses.
- Capital cost - insurance providers need to access on-demand finance to be able to cover payouts that exceed the premiums paid by insurance policyholders.
- Loss adjustment procedures - insurance providers need to take into consideration economic losses that result from covered risks.

Using index-based insurance reduces a number of the aforementioned costs. For example, loss adjustment procedures will not be required since the actual losses are not factored into the payout process (Burke et al., 2010). Moral hazard is greatly reduced since payouts are determined solely by objective weather elements that insurance policyholders will not have an influence on. Adverse selection is also greatly reduced as all farmers will receive the same payout rate irrespective of their individual risk profile and insurers no longer have to incur the cost of determining more risky farmers. Figure 2 shows a comparison of the costs associated with both traditional and index-based insurance products. From the figure, we can see that index-based insurance products are superior from a cost perspective as they exhibit fewer costs.

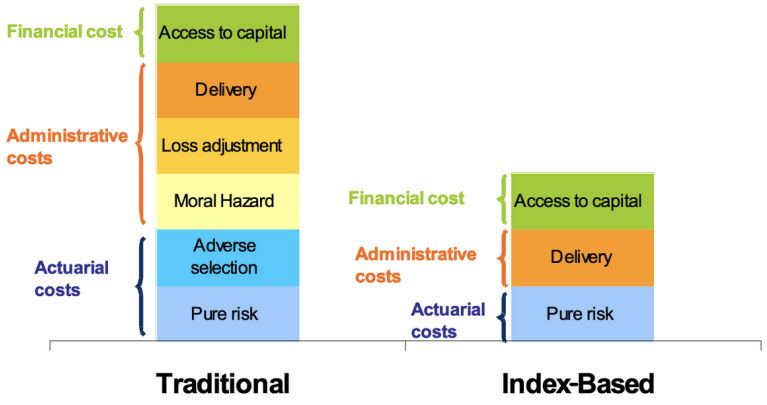

Figure 2: Comparing traditional insurance cost structure (left) and index-based insurance products cost structure (right). Index-based insurance products are superior from a cost perspective as they exhibit fewer costs. Source: (Burke et al., 2010).

2.3 Case Study: Agriculture and Agricultural Insurance in South Africa

South Africa has a largely dual agricultural system, split between large-scale commercial farming operations and small-scale farming [2]. Commercial farming operations have a major stack in the overall agricultural system and supply major food outlets, whilst small-scale farming operations are predominant in the rural areas of South Africa. Smallholder farmers can barely afford insurance and mostly incur huge losses in cases of adverse weather conditions such as drought due to only having access to limited resources and infrastructure. Problems faced by smallholder farmers include low literacy rates, limited access to finance, low technology uptake and limited record-keeping procedures (Mkuhlani et al., 2019). These problems inhibit the farmers' potential to produce to the best of their ability. Further, due to limited access to credit opportunities, farmers in emerging economies tend to be more risk-averse and hence uptake of modern technologies is very low in emerging economies including South Africa (Gine and Yang, 2009).

The Intergovernmental Panel's 2019 report [3] on climate change describes the Southern African region as a hotspot for projected climate change and many studies as cited in the report show that temperatures in this region are rising at twice the global rate leading to more frequent heatwaves. With these observations in mind effective agricultural insurance schemes are increasingly being needed more and more, hence the call for more cost-effective ways to insure smallholder farmers.

2.3.1 Current Agricultural Insurance Schemes in South Africa

Given that smallholder farmers earn low income and are vulnerable to weather hazards, weather index insurance can prove to be the missing risk management solution that they need (Tadesse et al., 2015). Despite the ever-increasing need for risk management mechanisms, in the South African context, weather index crop insurance remains unexplored and is generally not commercially available. Although there is not much index-insurance activity in South Africa, there has been significant progress in East Africa. For example, Etherisc [4], a company specializing in blockchain-based agricultural insurance solutions, has successfully piloted a parametric crop insurance platform built on the Ethereum blockchain in Kenya (Bhusal, 2021).

The insurance coverage for agriculture in South Africa is primarily categorized under the short-term or non-life insurance division of the insurance sector. One of the leading crop insurers in South Africa is Santam[5]. Although not involved in the index insurance space (to my knowledge), they follow the traditional route of in-field assessments by insurance agents to issue payouts. Other notable crop

[2] https://www.trade.gov/country-commercial-guides/south-africa-agricultural-sector
[3] https://www.ipcc.ch/srccl/
[4] https://etherisc.com/
[5] https://www.santam.co.za/products/agriculture/crop-insurance/

insurers in South Africa include Old Mutual[6], Hollard[7] and Agriseker,[8] all of which also follow the traditional route of in-field assessments. The South African Insurance Association (SAIA) is also playing a role in the insurance space by advocating for index crop insurance and holding workshops[9] in collaboration with the World Bank Group Global Index Insurance Facility (GIIF) to spread awareness of the potential benefits of weather index crop insurance. Studies conducted in several countries have proven the affordability and feasibility of such insurance mechanisms (Ghosh et al., 2020) hence the zeal to explore its practicality in the South African context.

2.3.2 Underwriting in Agricultural Insurance

Underwriting in agricultural insurance involves evaluating the risks of insuring a farmer's agricultural operations, determining the premiums to be paid as well as the terms and conditions to be included in the insurance policy and evaluating the risks involved, to aid in deciding whether to approve or decline an insurance policy (Kitur, 2018). Aggour et al. (2006), in their paper on automating the underwriting of insurance applications, outline the steps involved in the underwriting process, which include:

- Information gathering - the underwriter gathers information about the farmer. The information will include details about the agricultural operations that a farmer is currently engaged in, the agricultural assets being insured, farm location, crops being farmed and any claims or losses incurred in the past.
- Risk assessment - the underwriter performs a risk assessment to assess the risk of insuring a particular farmer's agricultural operations. The assessment takes into account factors such as the type of crops being insured, farm location, and the likelihood of incurring losses due to adverse weather conditions or other risks.
- Calculating the value of premiums - based on the assessed risk, taking into account the likelihood of incurring losses and the potential cost of those losses, the underwriter determines the premium for the insurance policy.
- Drafting the terms and conditions of the policy - the underwriter drafts the terms and conditions of the insurance policy including the coverage limits and any exclusions.
- Policy approval - if the risk is acceptable the underwriter approves the policy and issues it to the farmer. However, if the risk is perceived to be too high, the insurance policy may be offered either with higher premiums, more restrictive terms and conditions, or can even be declined.

[6] https://www.oldmutual.co.za/business/goals/run/agriplus/
[7] https://www.hollard.co.za/business-insurance/specialist-sector-insurance/agriculture
[8] https://www.agriseker.co.za/who-we-are/
[9] https://www.indexinsuranceforum.org/publication/agriculture-insurance-training-johannesburg

2.3.3 Challenges to Agricultural Insurance Uptake

Several limitations hinder the widespread adoption of agriculture insurance. These include inadequate understanding of insurance policies, the high cost of premiums, low agricultural productivity and income levels, the remote location of farms from insurance service providers, and negative attitudes towards insurance in general (Tsikirayi et al., 2013). Insurers attribute low uptake to farmers' limited incomes, while farmers point to the unaffordability of insurance products as the primary obstacle. With these challenges in mind, Tsikirayi et al. (2013) provides the following recommendations to improve uptake:

- Conducting research and development aimed at creating insurance products that are both cost-effective and that cater to the requirements of farmers, hence the work in this thesis.
- Educating farmers about the importance of agricultural insurance in their farming activities for example through extensive marketing campaigns by insurance companies.
- Raising agricultural production to create an agricultural pool that serves as the foundation for reasonably priced insurance premiums for example, through government intervention by way of subsidies to farmers.

2.3.4 Blockchain and Blockchain in Agriculture

A blockchain is essentially a decentralized database that stores data across a network of computers, making it near impossible to tamper with or manipulate the data once it has been recorded (Guo and Liang, 2016). One of the key features of blockchain technology is its use of cryptographic algorithms to secure the data stored on a network of nodes (Yaga et al., 2019). Each block in the chain contains a unique cryptographic hash that links it to the previous block, forming an unbreakable chain of data. Once data is recorded on a blockchain, it cannot be modified or deleted without the consensus of the network participants. This immutability ensures that data and transactions cannot be tampered with, which enhances the security of the system (Sakthi and DafniRose, 2022). Another important aspect of blockchain technology is its use of consensus mechanisms to validate transactions and maintain the integrity of the network. Different types of consensus mechanisms, such as Proof of Work and Proof of Stake, require participants in the network to solve complex mathematical problems or stake their own tokens to validate transactions and earn rewards. Each transaction is verified using digital signatures, which ensure that only the owner of the private key can initiate a transaction. This cryptographic security ensures that data and transactions are protected from unauthorized access and manipulation, which enhances the security and privacy of the system. Unlike traditional centralized systems, blockchain networks are decentralized, meaning that data and transactions are stored and processed across a network of nodes rather than a single point of control. This distributed nature of blockchain technology ensures that no single entity can manipulate or control data, providing a high

degree of transparency and security for data and transactions. This makes blockchains an attractive option for applications that require trust, accountability, and reliability (Algorand, 2023).

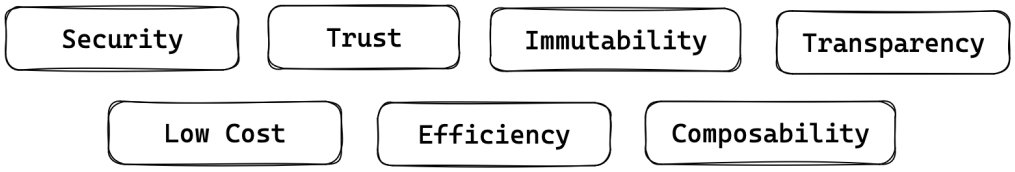

Figure 3: A list of features of blockchains that make them appealing for use in value-based applications. Source: (Algorand, 2023).

Blockchain technology is gaining popularity in various industries, including agriculture. One of the most significant applications of blockchain in agriculture is in supply chain management (Kanjere, 2021). It provides a secure, transparent, and decentralized way to manage data and transactions, which helps to improve efficiency, reduce costs, and enhance trust in agricultural supply chains. With blockchain, every step of the supply chain, from farm to fork, is recorded and tracked. This helps to ensure the authenticity, quality, and safety of agricultural products, as well as traceability in case of a recall. Farmers can record information about their crops, such as the date of planting, fertilization, and harvesting, on the blockchain. This information will be shared with stakeholders in the supply chain, such as a distributor or retailer, who can add their own information about the storage and transportation of the product. All parties are able to access and verify the information, which helps to prevent fraud, reduce waste, and increase efficiency.

Similar to its use in supply chains, blockchain technology is also used in building agricultural asset registries that allow farmers to access loans using their assets as collateral (Mzuku, 2019). An agricultural asset registry is a database that records the ownership and value of agricultural assets, such as crops, livestock, equipment, and land. By creating an asset registry on a blockchain, the ownership and value of assets can be recorded in a secure and transparent manner, which enables farmers to use their assets as collateral to obtain loans from financial institutions. Using their assets as collateral, farmers are able to access loans that they would otherwise be unable to obtain due to a lack of credit history or collateral. The information stored on a blockchain-based asset registry is verifiable, which in turn lowers the risk of fraudulent transactions and improves the accuracy of asset valuations. By providing a secure and transparent record of ownership and value, farmers can have more confidence in the lending process and financial institutions can have greater confidence in the collateral provided by farmers.

Another application of blockchain in agriculture is in the management of land and natural resources. In their paper on the role of blockchain in documenting land users' rights, Desiree and Chinwe (2020)

highlight that blockchain technology helps to ensure land ownership and usage rights are transparent and secure, reducing disputes over land ownership and misuse. It also facilitates the management of water resources, helping to ensure that water is used efficiently and sustainably (Sakthi and DafniRose, 2022).

2.3.5 Challenges and Opportunities that Blockchain Technology Presents in Agriculture

A potential challenge for blockchain-based weather index crop insurance platforms is the need to navigate the complex regulatory landscape. The regulatory landscape for weather index insurance varies by country and region, with different regulatory frameworks in place for different types of insurance products. In general, insurance is a heavily regulated industry, with regulatory agencies responsible for overseeing the activities of insurers and protecting policyholders from financial harm. In South Africa, the regulatory landscape for weather index crop insurance is governed by the Financial Sector Conduct Authority (FSCA)[10], which is the primary regulator for the financial services industry. The FSCA is responsible for regulating the activities of insurers and protecting policyholders from financial harm. In some cases, existing insurance regulations may not be well-suited to the unique features of blockchain technology, such as decentralization and smart contracts. This can create regulatory hurdles for blockchain-based weather index crop insurance platforms, particularly in terms of obtaining licenses and approvals from regulatory agencies. A weather index crop insurance platform needs to obtain licenses and approvals from FSCA to operate legally. This process can be complex and time-consuming. It may also require the submission of detailed business plans and other supporting documentation. In addition, blockchain-based weather index insurance platforms need to comply with a range of other regulations, such as those related to data and consumer protection eg. the Protection of Personal Information Act (POPIA), and custody of digital assets.

On the other hand, blockchain technology also presents some opportunities for regulatory improvement in the insurance industry in South Africa. For example, the transparency and immutability of blockchain-based systems can make it easier for the FSCA to monitor the activities of insurers and ensure that they are complying with laws and regulations. In addition, the use of smart contracts can automate the execution of insurance contracts and reduce the need for manual underwriting and claims processing, which could improve the efficiency and effectiveness of the regulatory process.

The regulatory landscape for weather index insurance in South Africa is complex and evolving, and blockchain-based weather index insurance platforms will need to navigate this landscape carefully in order to be successful. While there may be challenges to overcome, there are also opportunities for regulatory improvement and innovation through the use of blockchain technology hence, making blockchain-based weather index crop insurance viable for exploration.

[10] https://www.fsca.co.za/Pages/About-Us.aspx

2.3.6 Landscape for Innovation in Agricultural Insurance in South Africa and Africa

South Africa and Africa as a whole present a unique landscape for innovation in agricultural insurance. The continent's largely agrarian economies heavily depend on agriculture, which provides livelihoods for the majority of the population. However, agricultural productivity is often constrained by a range of factors, including climate change which in recent years has seen an increase in the frequency of heat waves causing severe droughts, pests, diseases, and lack of access to finance and insurance (Iizumi and Ramankutty, 2015).

There is a significant opportunity for innovation in insurance to help smallholder farmers manage these risks and protect their livelihoods. South Africa, in particular, has a well-established insurance industry, with a range of products and services available to firms and households. However, agricultural insurance products are often prohibitively expensive for smallholder farmers, who may lack the resources and information needed to access them. To address these challenges, there is a need for innovative approaches to agricultural insurance that are tailored to the needs of smallholder farmers in South Africa and Africa. According to Mapfumo (2007), one promising approach is the use of weather index-based crop insurance, which relies on weather and other related factors to trigger insurance payouts to farmers. This approach as earlier alluded, can be more cost-effective and reliable than conventional crop insurance, which requires expensive and time-consuming assessments of crop losses.

The development of weather index insurance products requires accurate and timely weather data, which may require investment in weather monitoring systems and other infrastructure. To ensure affordability there is a need to research cost-effective ways of collecting weather data so as to make the system affordable to small-scale farmers. Similarly, the success of agricultural insurance products depends on farmers' understanding of the products and their ability to access them, which may require investment in education programs and the development of intuitive insurance platforms that are easy to understand and use (Weber, 2019). In addition, innovation in agricultural insurance can be supported by advances in technology, such as blockchain, which can improve data transparency and reduce the potential for fraud. By leveraging these technologies, insurance companies can provide more transparent and efficient services, while also reducing transaction costs and increasing access to insurance products.

Considering all these possibilities, there is a case for an exploration of the viability of a blockchain-based weather index crop insurance platform specifically tailored for small-scale farmers to provide low-cost, accessible and transparent crop insurance in South Africa and Africa at large, hence this research project.

3 Design And Methodology

In this section, I outline the development approach, methodology and process of gathering requirements that I utilise to guide the design and implementation of the weather index crop insurance platform.

3.1 Development Approach

To develop the weather index crop insurance platform, I use an agile development[11] approach in which I break down the project into smaller, manageable tasks and constantly iterate on each component to ensure that the project's objectives are being met. To ensure that the platform is user-friendly and efficient, I use design thinking[12] and a user-centric design[13] approach. This involves conducting research and analysis on the current insurance platforms, and their shortcomings to establish the needs and requirements that the new platform seeks to solve for smallholder farmers. To guide the development approach, I use the principle of continuous improvement and by embracing this value, a robust and effective weather index crop insurance platform that meets the needs of the intended users is developed.

3.2 Requirements Specification

To develop the blockchain-enabled weather index crop insurance platform, it is crucial to define and articulate the key requirements necessary to meet the needs of smallholder farmers and to address the challenges they face with regard to fully utilising the services of conventional insurance platforms. In this specification, I outline the features, functions and capabilities of the platform being developed. The outline serves as a blueprint for the development of the platform to ensure the resulting product aligns with the requirements. I analyse the current insurance platforms to identify the key pain points and challenges being faced by smallholder farmers in relation to weather risks, crop loss, insurance coverage and claim processing. The resulting key areas of focus for the new platform include the user interface, weather data collection and payout processing.

3.2.1 Functional Requirements

I list the functional requirements (FR*) below:

- **FR1** - The platform should allow for weather data management - weather data capture using IoT sensors, storage of the weather data on-chain and publishing of the data via an application

[11] https://www.atlassian.com/agile/project-management
[12] https://online.hbs.edu/blog/post/what-is-design-thinking
[13] https://online.hbs.edu/blog/post/what-is-human-centered-design

programming interface (API). The data should be relevant to the farmer's field location and crops.

- **FR2** - Digital insurance policy management - the process of insurance policy creation and farmers joining policies should be managed via the platform.
- **FR3** - The platform should have an automated payout system on-chain that is secure and reliable. Insurance payouts should be triggered by strike conditions configured per insurance policy.
- **FR4** - Users should be able to easily make premium payments and receive payouts.
- **FR5** - The system should have access control for sensitive client details.

3.2.2 Non-functional Requirements

I list the non-functional requirements (NFR*) below:

- **NFR1** - The platform should be user-friendly and not too technically complicated.
- **NFR2** - Users should be able to access relevant information and features with minimal effort.
- **NFR3** - The proposed system should have a reliable storage system where all processing information can be stored.
- **NFR4** - The platform should be responsive and accessible from various devices, including desktop and mobile.
- **NFR5** - There should be minimum latency in information propagation in the system.
- **NFR6** - The proposed platform should process data accurately, quickly and automatically.
- **NFR7** - Processes like calculations should be done efficiently without errors.

3.3 Platform Wireframes

To establish the layout of screens and their flow for task completion, I utilise wireframes[14]. One major benefit of wireframes is that they allow for the overall design of the platform to be laid out without one getting bogged down with details such as specific color schemes or visual graphics. They allow for the establishment of a clear and intuitive layout and flow of application screens. Wireframes allow for experimentation with different layouts and flows before a commitment is made to a specific design. During wireframing, I place focus on creating a clear and intuitive user experience and I also consider factors such as content hierarchy as well as the flow of tasks that the users need to complete. Wireframes are an invaluable tool in the design process. Figure 4 shows the platform's wireframes showing the four main web pages of the platform - from left to right - (i) the farmer account details page; (ii) the insurance policy joining page; (iii) the insurance payouts page; and (iv) the sensor data page.

[14] https://protoio.medium.com/why-wireframes-are-important-in-the-design-process-de4e773e611

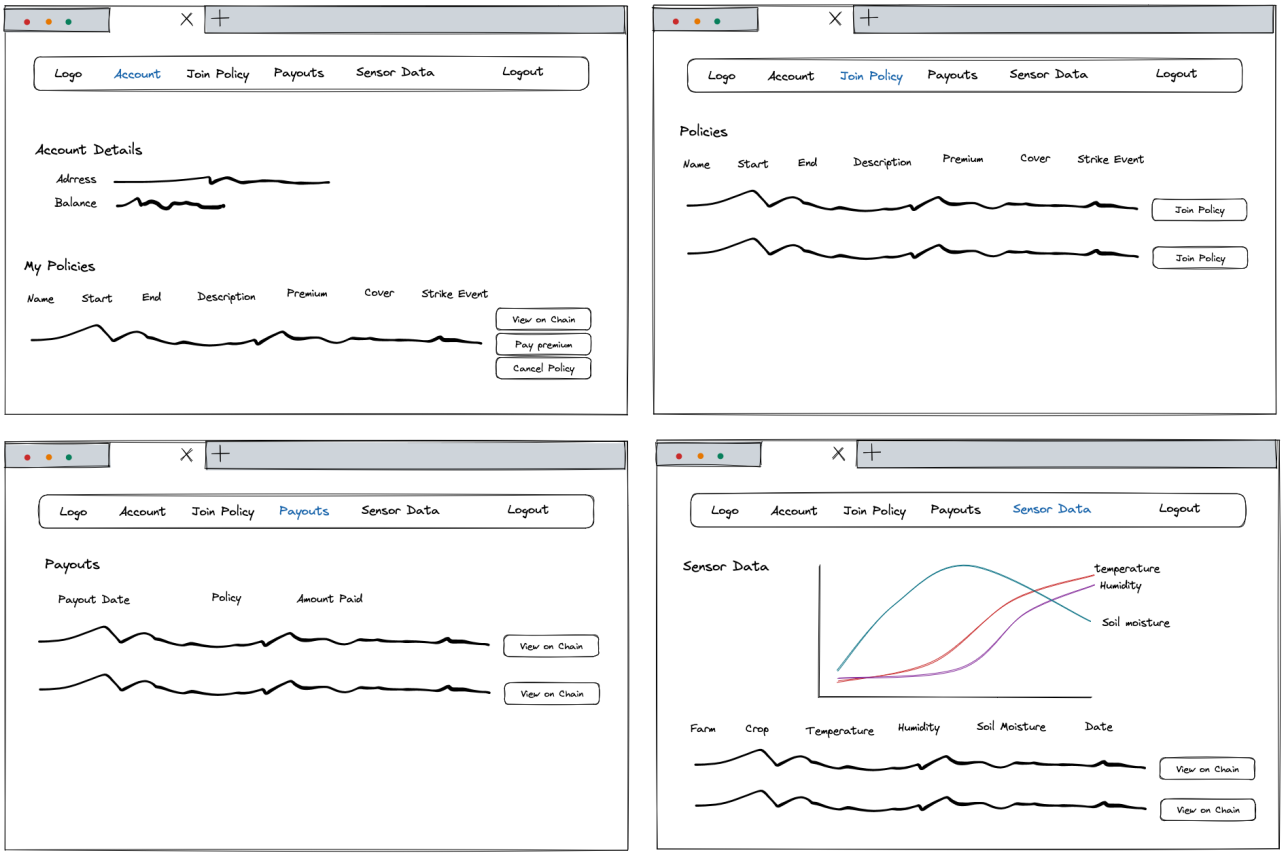

Figure 4: Platform wireframes showing the four main web pages of the platform - from left to right - (i) farmer account details page; (ii) insurance policy joining page; (iii) insurance payouts page; and (iv) sensor data page.

3.4 Weather Data Collection System

For weather data collection in this project, I design a low-cost temperature, humidity and soil moisture sensor reading prototype system using a DHT11 humidity and temperature sensor, a capacitive soil moisture sensor and an ESP32 microcontroller.

The hardware components are as follows:

- A DHT11 humidity and temperature sensor.
- A capacitive soil moisture sensor.
- An ESP32 microcontroller development board.
- A breadboard to connect all the components and build the circuit.
- Jumper wires to connect the sensors and ESP32 microcontroller on the breadboard.

3.4.1 DHT11 Humidity and Temperature Sensor

The DHT11 humidity and temperature sensor is a low-cost device that can measure both the humidity and temperature of an environment. It uses a capacitive humidity sensor and a thermistor to measure these values, and it can communicate with a microcontroller or other devices via a single digital data pin. The DHT11 sensor has a range of 20-90% for humidity readings and 0-50°C for temperature readings, with an accuracy of +/- 5% for humidity and +/- 2°C for temperature (Components101, 2023).

Some of the advantages of DHT11 humidity and temperature sensors are:

- Easy to use, widely available and affordable.
- The sensor operates at a low voltage and has low power consumption, making it suitable for battery-powered devices.
- Compact and lightweight making it easy to integrate into projects with limited space.

Figure 5 shows an image of a typical DHT11 humidity and temperature sensor and Table 1 provides a description of the pinout structure and the function of each pin.

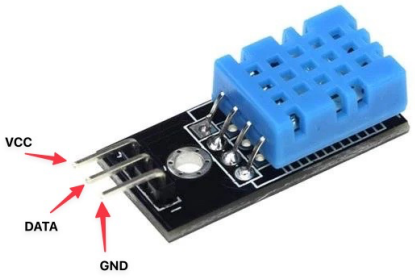

Figure 5: DHT11 humidity and temperature sensor structure and pinout labels. VCC and GND pins power up the sensor. The DATA pin provides the digital temperature and humidity output reading. Source: (Components101, 2023).

DHT11 pin	Description
VCC	Power supply 3.5V to 5.5V
GND	Ground
DATA	Serial temperature and humidity reading output

Table 1: DHT11 humidity and temperature sensor pinout reference sheet listing the pins available on the DHT11 sensor and the description of each pin. Source: (Components101, 2023).

3.4.2 Capacitive Soil Moisture Sensor

Capacitive soil moisture sensors measure the water content of soil by detecting changes in the soil's dielectric constant (ability to store electrical energy in an electric field) (Circuitschools, 2023). The sensor typically consists of two parallel metal electrodes, one of which is inserted into the soil and the other is used as a reference electrode. The electrodes are connected to a circuit that can measure the capacitance between the electrodes. The capacitance of the soil changes as the water content of the soil changes. When the soil is dry, the capacitance is low, and when the soil is wet, the capacitance is high. By measuring the capacitance of the soil, the sensor can accurately determine the water content of the soil. The capacitive soil moisture sensor is relatively easy to use and can be connected to a microcontroller or data logger to record and monitor soil moisture.

Some of the advantages of capacitive soil moisture sensors are:

- High accuracy and stability.
- Can be used in a wide range of soil types.
- Can be used in both indoor and outdoor environments.
- Low power consumption.

Figure 6 shows an image of a typical capacitive soil moisture sensor and Table 2 provides a description of the pinout structure and the function of each pin.

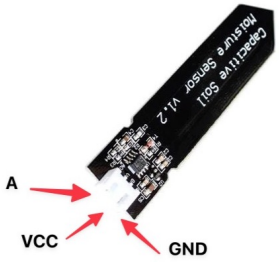

Figure 6: Capacitive soil moisture sensor structure and pinout labels. VCC and GND pins power up the sensor. The A pin provides the soil moisture reading as analog voltage output. Source: (Circuitschools, 2023).

Soil moisture sensor pin	Description
VCC	Power supply 3.5V to 5.5V
GND	Ground
A	Capacitive soil moisture analog output

Table 2: Capacitive soil moisture sensor pinout reference sheet listing the pins available on the soil moisture sensor and the description of each pin. Source: (Circuitschools, 2023).

3.4.3 ESP32 NodeMCU Microcontroller

The ESP32 NodeMCU is a development board based on the ESP32 microcontroller, a low-cost, low-power system-on-a-chip (SoC) with WiFi and Bluetooth capabilities (Espressif, 2023). The NodeMCU board is designed to make it easy to work with the ESP32, and it includes a USB-to-serial converter, a power supply, and a set of general-purpose input/output (GPIO) pins that can be used to connect to various sensors. The ESP32 microcontroller on the NodeMCU board features two 32-bit cores, each running at up to 240 MHz. It also includes a wide range of peripheral interfaces, such as the Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), Universal asynchronous receiver-transmitter (UART), and Pulse-width modulation (PWM), which can be used to connect to various sensors and devices. Additionally, the ESP32 has built-in support for WiFi and Bluetooth, which makes it ideal for a wide range of IoT applications. The ESP32 NodeMCU board supports MicroPython, an open-source programming language that is designed for microcontrollers, which makes it easy to program the ESP32 using the Thonny IDE for MicroPython. Because the board has WiFi capabilities the data collected from sensors can be transmitted over a network allowing for remote monitoring.

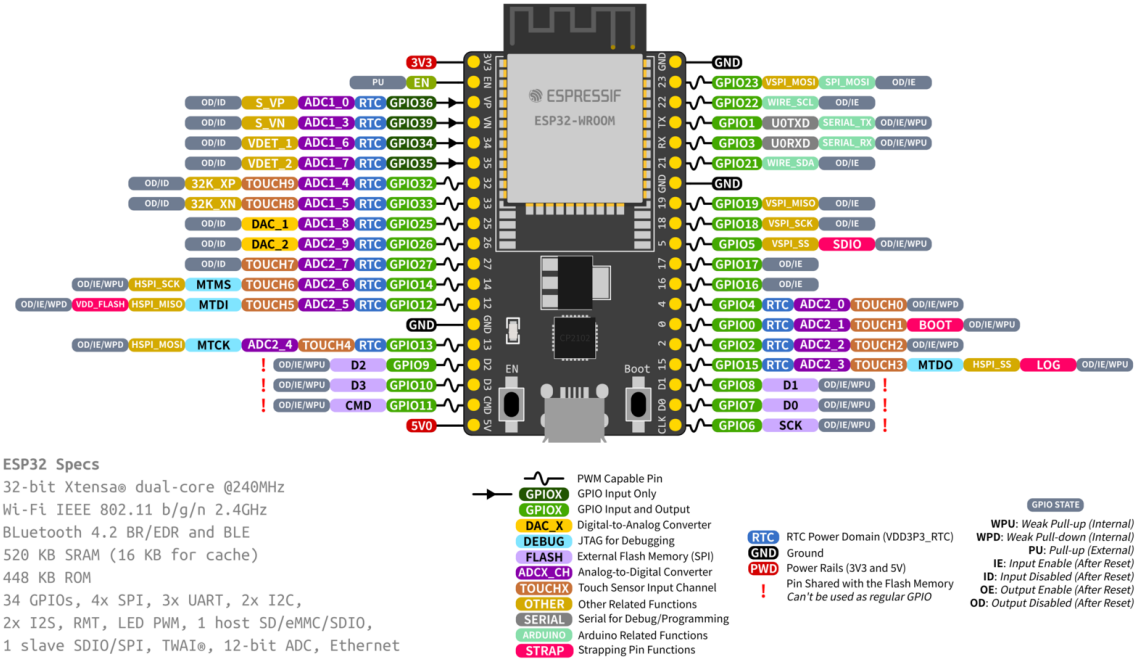

Figure 7: ESP32 NodeMCU microcontroller structure and pinout labels. The microcontroller collects humidity and temperature readings from a DHT11 humidity and temperature sensor connected to digital pin 5. Soil moisture readings are collected from a soil moisture sensor connected to analog to digital pin 34. The sensors are powered by connecting their VCC and GND pins to the microcontroller's 3.3V and GND pins respectively. Source: (Espressif, 2023).

In the weather data collection circuit design for this platform, the three pins on the DHT11 humidity and temperature sensor are connected via jumper cables to the microcontroller. VCC to 3.3V, Data to digital pin 5 and GND to GND. Similarly, for the soil moisture sensor, GND is connected to GND, VCC to 3.3V and A to an analog to digital pin 34 on the ESP32 board. See Appendix A for circuit connections. Using MicroPython code[15], the ESP32 board is programmed to connect to WiFi and to read weather data from the sensors at regular intervals of 2 hours. The DHT11 humidity and temperature sensor, capacitive soil moisture sensor, and ESP32 microcontroller are to be deployed in the field to collect data on temperature and soil moisture levels. The collected data is then sent to an application programming interface (API)[16] which stores the data in a database and logs it to the Algorand blockchain.

3.5 Data Storage and Database Design

Data storage and database design are crucial components of the weather index crop insurance platform as there is a need for secure storage and management of sensitive data such as the transaction history of payouts. To ensure immutability and transparency, platform data is stored on the Algorand blockchain. The use of a blockchain provides a decentralized and secure way to store data. Each transaction is recorded in a block, which is then added to the chain, forming a tamper-proof and transparent ledger. In this application, the Algorand blockchain is used to store the weather data that trigger insurance payouts, as well as the information about which policy each farmer has subscribed to and to facilitate payout transactions.

However, storing and retrieving all platform data to and from the blockchain can be slow and expensive due to transaction validations and fees. To cater for this, a PostgreSQL database is also used for redundancy and to store additional data used in the system. PostgreSQL[17] is a reliable, scalable, and open-source relational database system that can handle complex queries and ensure data consistency. The main entities that I use to design the database schema in this application are users, roles, policies, strike events, weather data, premium payments and payouts. Insurance policies on the platform are crop-specific and are associated with a single crop, with a validity period denoted by a start and end date and covering a specific geographical location. Each user can join policies, and each policy is associated with a specific trigger weather event. When the trigger weather event occurs, the insured user is eligible for a payout. In terms of database design, separate tables are created for each entity and relationships are established between the entities using foreign keys. Figure 8 shows the outline of the database schema design, showing the relevant entities on the platform, their attributes and relationships. The schema also shows the mapping tables for the relationships between

[15] https://github.com/rideam/bima/tree/main/microcontroller
[16] https://bima.onrender.com/weather
[17] https://www.postgresql.org/

the entities.

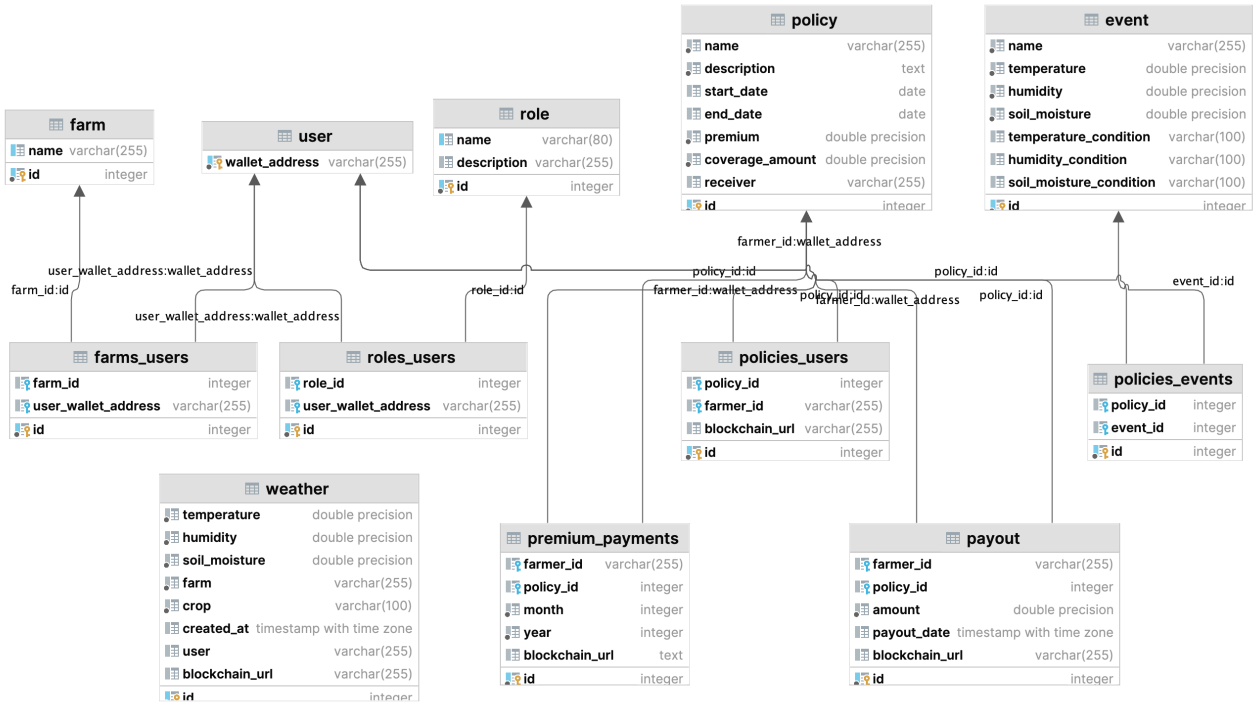

Figure 8: Database schema showing the relevant entities, their attributes and relationships on the platform. The main entities on the platform are users, roles, policies, strike events, weather data, premium payments and payouts. The schema also shows the mapping tables for the relationships between the entities.

3.5.1 Why Algorand Blockchain

The blockchain of choice for developing the blockchain-enabled weather index crop insurance platform is Algorand. Algorand is a blockchain platform designed to be secure, scalable, and decentralized which aims to solve some of the most pressing challenges faced by other blockchain platforms, such as slow transaction speeds, high transaction fees, and limited scalability (Chen and Micali, 2019). At its core, Algorand uses a Pure Proof of Stake (PPoS) consensus mechanism that ensures fast and secure transaction processing. Unlike other Proof of Stake blockchains that use delegated staking or other forms of centralization, Algorand's consensus mechanism is fully decentralized, meaning that anyone can participate in the consensus process and earn rewards for validating transactions (Chen and Micali, 2019). Algorand's consensus mechanism also ensures that the platform is highly secure and resistant to attacks (Micali, 2016). The platform's consensus protocol randomly selects a small group of users, known as a committee, to validate transactions and add new blocks to the blockchain. The committee members are chosen at random from a large pool of stakers, making it

extremely difficult for attackers to gain control of the network[18]. The native asset used to facilitate transactions on the Algorand blockchain is the ALGO[19].

In addition to its secure and scalable consensus mechanism, Algorand also offers a range of other features and benefits crucial to the development of the blockchain-enabled weather index crop insurance platform. These include:

- High transaction speeds - Algorand blockchain is capable of processing approximately 6000 transactions per second[20]. This is a considerably high throughput in comparison to other blockchain platforms like Ethereum at approximately 20 transactions per second[21]. The high throughput of the Algorand blockchain guarantees that the weather index crop insurance platform will not throttle as the volume of transactions increases.
- Low transaction fees - Algorand's fixed transaction fee of 0.001 ALGO [22] is significantly lower than other blockchain platforms, making it an attractive option to use for the blockchain-enabled weather index crop insurance platform where I am looking to minimise costs to make the platform affordable for smallholder farmers.
- Developer tools - Algorand offers a range of developer tools, including SDKs[23] for popular programming languages, APIs, and developer documentation[24], making it easy to build decentralized applications on the platform.
- Decentralized finance (DeFi) support - Algorand supports a range of DeFi applications, including stablecoins, decentralized exchanges, and other financial applications. This opens up the possibility of enhancing the product offering of the platform.

[18] https://developer.algorand.org/docs/get-started/basics/why_algorand/#the-consensus-protocol
[19] https://www.algorand.foundation/the-algo
[20] https://developer.algorand.org/docs/get-started/basics/why_algorand/#throughput
[21] https://crypto.com/university/blockchain-scalability
[22] 0.001 ALGO transaction fee equates to about ZAR 0.002 and is significantly lower compared to the average transaction fee of Ethereum which is 0.0009 ETH and equates to ZAR 30.47 as at 06/08/2023.
[23] https://developer.algorand.org/docs/sdks/
[24] https://developer.algorand.org/docs/

4 Implementation

In this section, I outline the features that I implement in the weather index crop insurance platform as well as the tools and frameworks I use in the development process.

4.1 Technology Stack

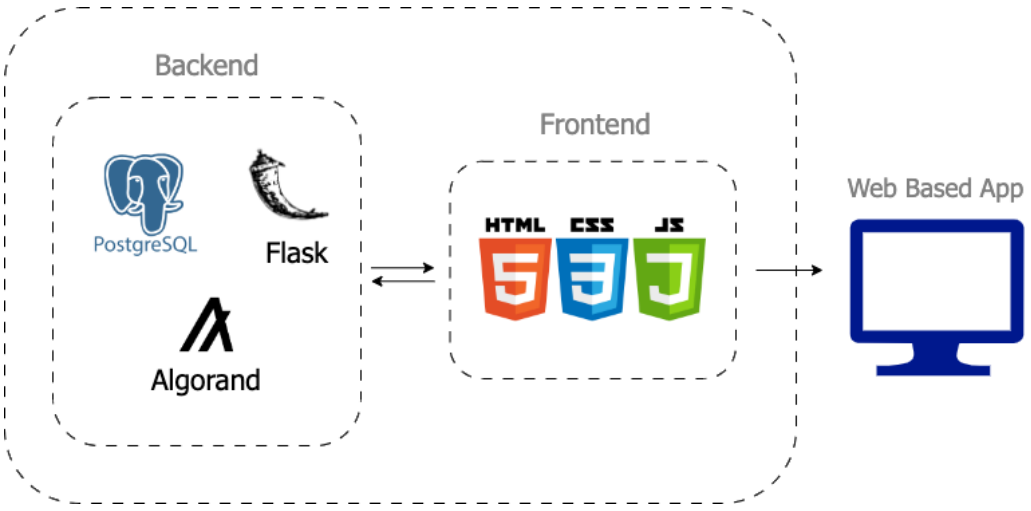

Figure 9: Components that make up the weather index crop insurance platform. The resulting platform is a web-based application built with the Flask framework. Weather and transactional data is stored on the Algorand blockchain as well as in a PostgreSQL database.

The developed weather index crop insurance platform is a web-based application that can be accessed from any device that has an internet connection. The application uses Flask, a microweb framework that helps developers create web applications using Python, JavaScript, HTML, and CSS. I utilise the Flask framework to develop the project because of its simplicity, flexibility, and scalability. Allowing for quick and efficient development of the platform with minimal boilerplate code. To simplify the user interface and make the platform mobile-friendly, I use the Bootstrap framework which provides ready-made web components and styling. To store the application's data, I use two technologies: the Algorand blockchain network and a PostgreSQL database. The Algorand blockchain network is a decentralized public blockchain that provides a secure and transparent way to store data. It uses a consensus mechanism called Pure Proof of Stake (PPoS) to verify transactions and secure the network. I used it in this project because of its immutability, to allow for fast and easily verifiable payment transactions and weather data storage. To write data to the blockchain and process payment transactions, I make use of the Algorand Python Software Development Kit (SDK)[25], a Python library

[25] https://github.com/algorand/py-algorand-sdk

that makes it easy to interact with the Algorand blockchain network without using smart contracts. PostgreSQL is a powerful open-source relational database management system that provides reliable and efficient data storage. It is widely used in web application development due to its robustness, scalability, and support for complex queries. I use it in this project to allow for quicker retrieval of weather data to be displayed on the platform. I use Git and GitHub for version control and the project source code is available on GitHub[26]. To program the ESP32 microcontroller for weather data collection and transmission, I use the Thonny IDE and MicroPython[27].

4.2 System Architecture

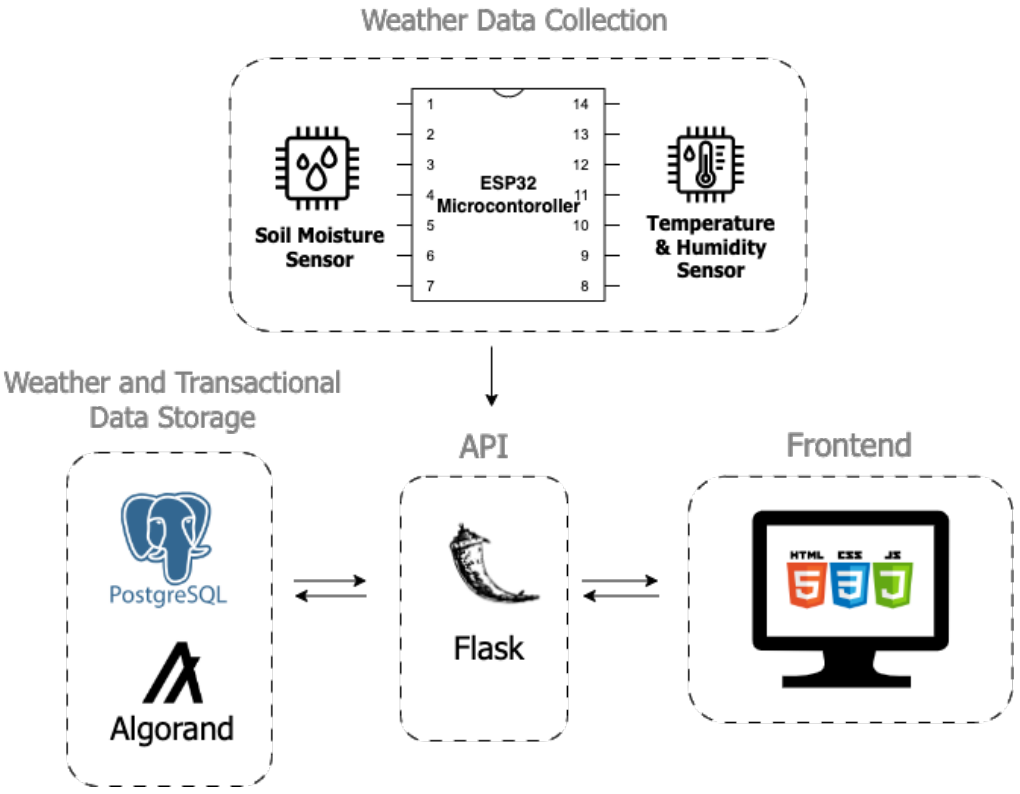

Figure 10: System architecture showing a 4-tier architecture - (i) the frontend; (ii) the API; (iii) the weather and transactional data storage; and (iv) the weather data collection layers. Weather data is collected from sensors attached to a microcontroller. The microcontroller posts the weather data to the API which coordinates the storage and retrieval of the data to be displayed on the frontend webpages.

[26] https://github.com/rideam/bima
[27] https://github.com/rideam/bima/tree/main/microcontroller

The platform follows a 4-tier architecture, with a frontend, API, weather data collection and data storage layer. The frontend is implemented using Bootstrap, while the backend is built using Flask, and the data storage layer is powered by both PostgreSQL and the Algorand blockchain. Sensors connected to a microcontroller handle weather data collection. The frontend is further divided into two components: a farmer portal and an administration portal. The farmer portal allows farmers to join insurance policies, pay premiums, see the weather data collected from their farm and view a record of insurance payout transactions. The administration portal allows the system administrator to register farm information, enter insurance policy and strike event details and perform other record manipulation tasks such as updating records. The backend is composed of three components: (i) PostgreSQL database that stores platform data off-chain for faster retrieval; (ii) Algorand blockchain that stores the weather data and transactions on-chain for immutability and verifiability; and (iii) Flask app that handles the logic for registering farmers, saving weather data, insurance policy data and data about transactions on and off-chain. The developed platform is database agnostic - the application uses SQL Alchemy an object-relational management system that makes it database agnostic with any relational database system such as PostgreSQL or MySQL.

4.3 Implemented Features

The features I implement in the application are as follows:

Farmer portal:

- Farmer account creation and authentication - upon account creation a farmer receives a 25-word passphrase which they can use to log in to the system.
- Farmer account details page - upon successful login a farmer is able to view their account details including but not limited to their ALGO balance, their wallet address and the policies they have joined.
- Crop insurance policy details page - farmers are able to view the list of available policies on the policy details page and join a desired insurance policy.
- Insurance payouts page - farmers can view a list of payouts that were issued when strike event conditions in their policy were triggered by extreme weather data recorded from sensors.
- Sensor data page - farmers can view the weather data collected by sensors both as a graph and numerical representations of the records.
- Blockchain integration - weather data from sensors, insurance policy premium payment transactions and insurance payout transactions are processed and stored on the Algorand blockchain for immutability and verifiability.

Administration portal:

- System administration - the system administrator through the administration portal is able to add information about policies and strike events as well as manage farmer information such as farm details.

4.3.1 User Interface Implementation

I build the user interface using the design phase wireframes and design specifications, see Section 4.6 for more details on the user interface. I use Bootstrap to implement a responsive and user-friendly interface that enables users to interact with the platform seamlessly on different device form factors such as desktop and mobile. There are two main users of the platform as depicted in Figure 11, showing the actions they can perform on the platform: a farmer and a system administrator.

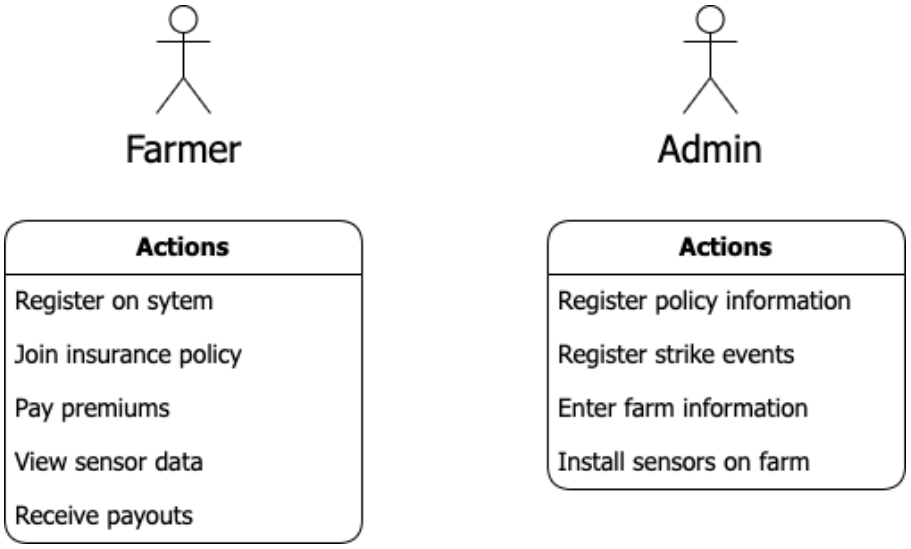

Figure 11: Types of users on the platform and the actions they can perform. The platform primarily serves two users: farmers, who receive insurance services, and system administrators, who handle administrative tasks such as adding new policy details onto the platform.

The workflow of the two users in the system is depicted in Figure 12, showing the actions users can perform and the flow of data on the platform:

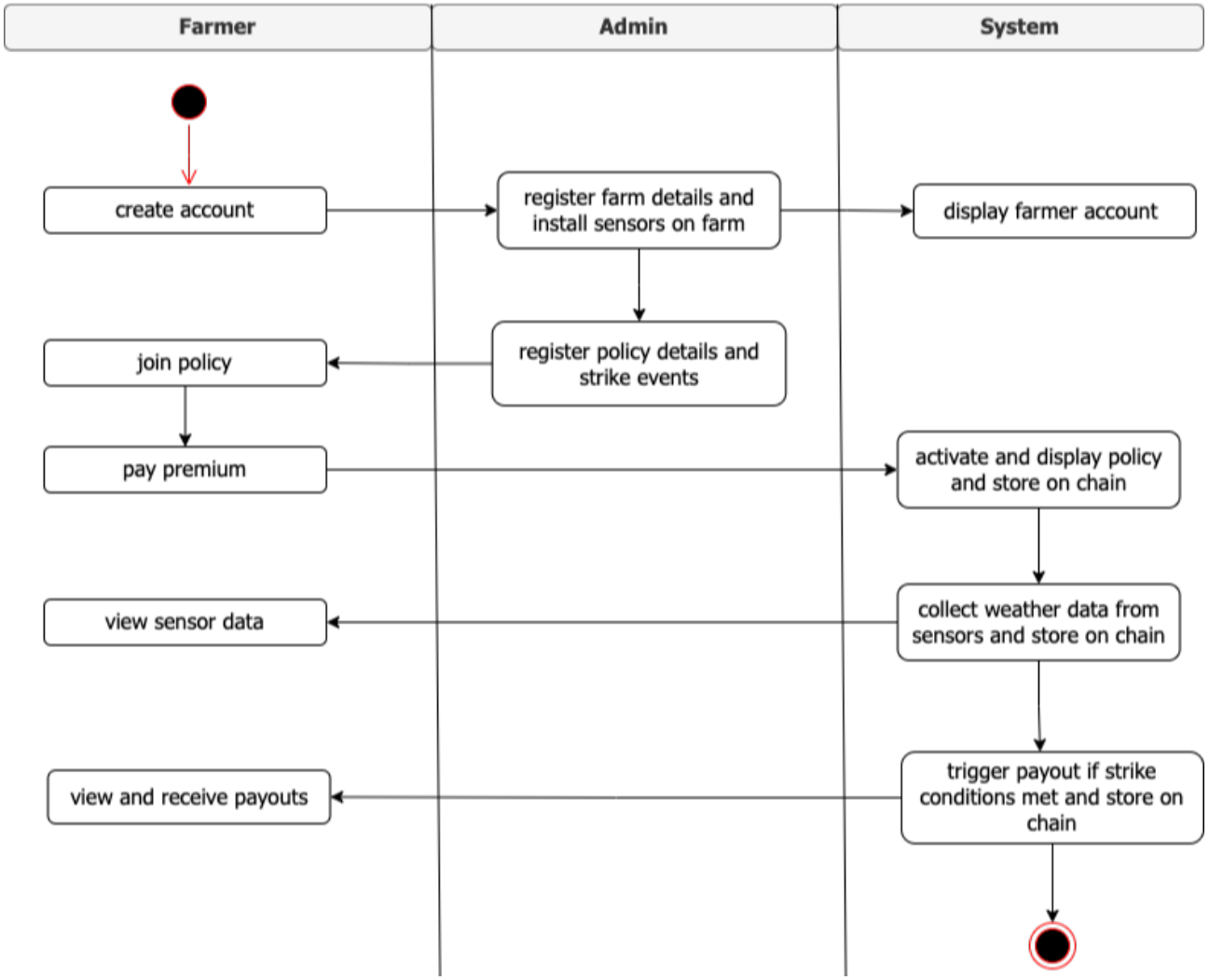

Figure 12: System workflow diagram showing the actions platform users (farmers and system administrators) can perform and the flow of data on the platform. A farmer begins by creating an account on the platform. System administrators will then install sensors on the farmer's farm, and register the farm's details as well as insurance policy details. After registration, a farmer joins a policy. To activate the policy a farmer pays a premium. On the platform, a farmer is able to see the weather data collected from the sensors on their farm as well as the payout transactions triggered when strike conditions attached to their policy are met.

4.4 Platform Deployment

The application is deployed on Render[28] which offers cloud application hosting. The platform is named Bima, a Swahili word that translates to insurance.

[28] https://bima.onrender.com

4.5 Platform Details

In this section, I outline the typical workflow on the platform from farmer registration through to a farmer receiving a payout. Figure 13 provides an illustration of the process flow within the Bima insurance platform.

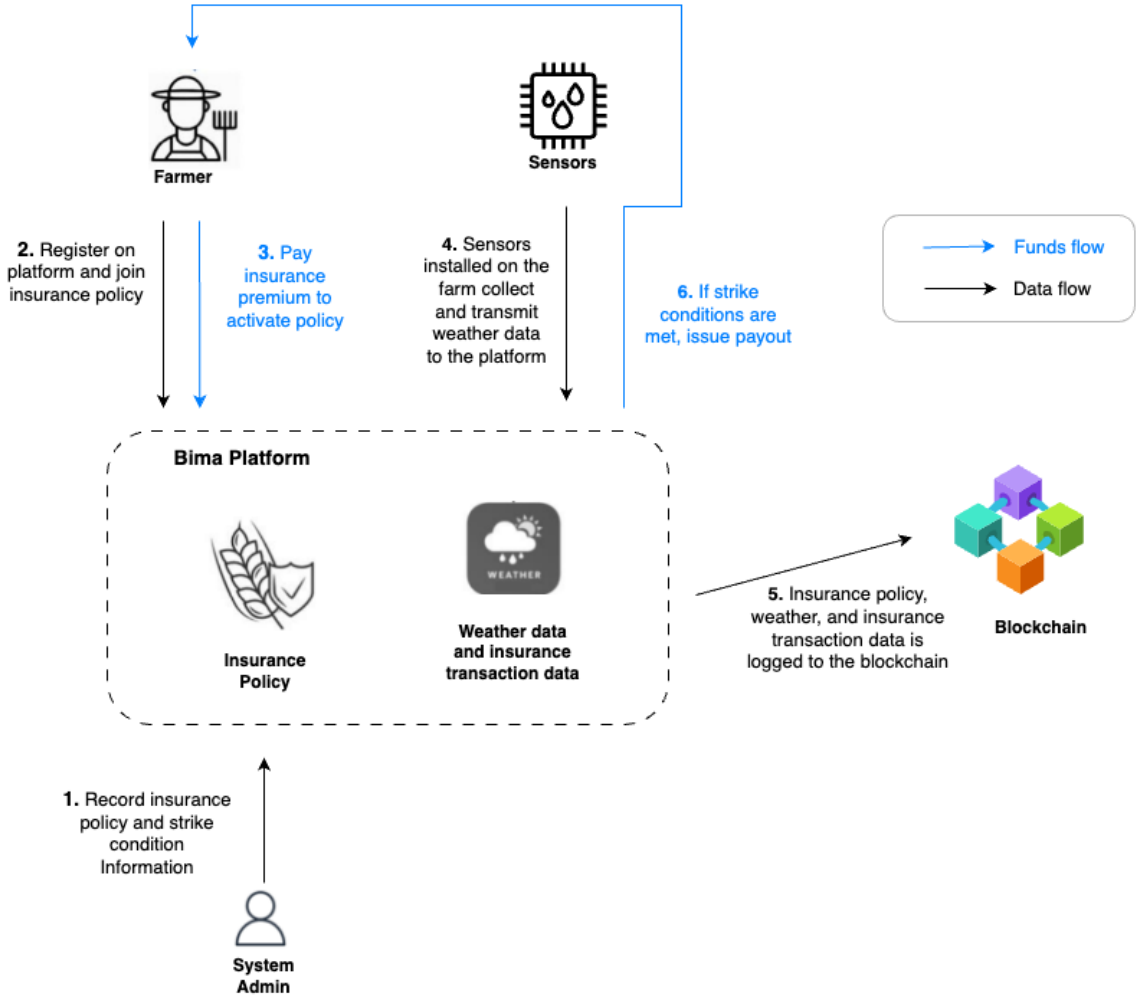

Figure 13: Illustration of the process flow within the Bima insurance platform. The flow begins with the administrator recording the insurance policy and strike condition information on the Bima platform (Step 1). A farmer then registers on the platform and joins the insurance policy (Step 2), and then proceeds to make payment of the insurance premium to activate the insurance policy (Step 3). Weather data is collected by sensors installed on the farm and sent to the Bima platform (Step 4). The insurance policy, along with weather and insurance transaction data, is logged onto the blockchain (Step 5). Finally, if the predefined strike conditions attached to the insurance policy are met, the platform issues an automatic and instant payout to the farmer (Step 6).

4.5.1 Farmer Registration and Insurance Policy Membership

Farmers register on the platform by clicking the *Create account* button found on the login page of the application. Upon account creation, they will receive a 25-word mnemonic that they can use to log in to the application. To start transacting on the platform a farmer needs to fund their account with ALGO. After registering on the platform the system administrator will then install sensors on the farmer's field and save the farm information on the platform as well as program the microcontroller with the identification information of the farm so that when it sends data to the API, the platform can identify from which farm the data is coming from. A farmer is able to join an insurance policy from the listed policies that an administrator would have entered on the platform. After joining a policy, a farmer needs to pay a premium each month to activate their joined policy. The transactions of farmers joining a policy are stored on the blockchain as well as other transactions like payment of premiums and insurance payouts. Figure 14 provides an illustration of a farmer joining an insurance policy on the platform and paying the corresponding premium to activate the policy.

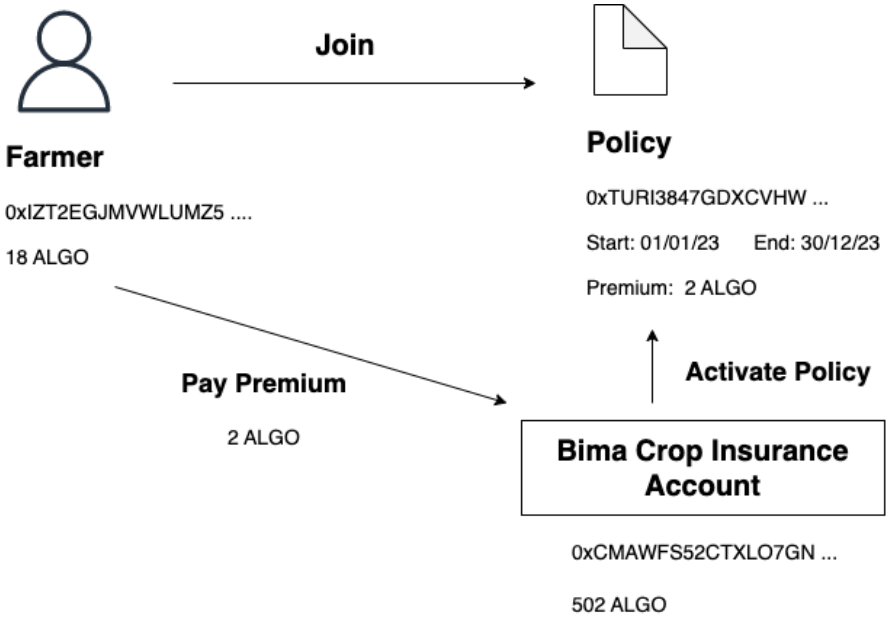

Figure 14: Illustration of a farmer joining an insurance policy and paying the corresponding premium to activate the policy. A farmer joins an insurance policy by clicking the join button on the platform. To activate the policy a farmer is required to pay a monthly premium stipulated in the insurance policy details. The premium is paid to the platform's Algorand account.

4.5.2 Weather Data Collection

Sensors attached to a microcontroller installed on the farmer's farm by an administrator send weather data readings to the platform. A farmer is able to view this weather data on the sensor data page of the platform. See Appendix A, which shows the weather data collection components and data logging process. Figure 15 provides an illustration of weather data being collected and stored on the insurance platform's Algorand blockchain account.

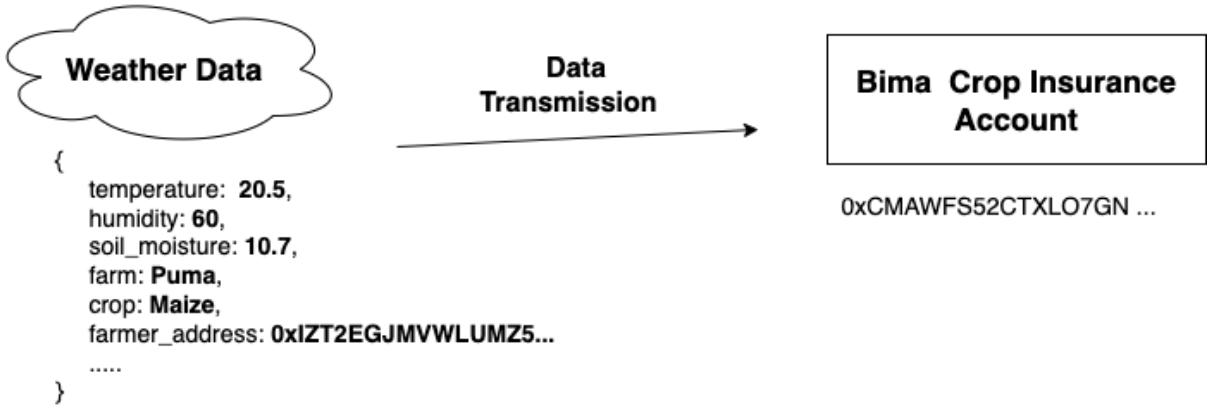

Figure 15: Illustration of weather data being collected and stored on the insurance platform's Algorand blockchain account. Weather data collected from sensors attached to an ESP32 microcontroller is sent to the platform which stores the data on the Algorand blockchain.

4.5.3 Triggering of Payouts

Payouts on the platform are triggered via the weather data API endpoint[29] which is the endpoint where the microcontroller sends sensor data to. The components of the post request are the values from sensors namely, temperature, humidity and soil moisture percentage, together with three other identifiers, the name of the farm where the sensors are installed, the crop, and the wallet address of the farmer. The identifiers are used to retrieve the currently active policy for the farmer which then allows the endpoint to check the strike conditions attached to the policy of the farmer. The temperature, humidity and soil moisture values are compared with the conditions of the strike event attached to the identified farmer's policy. If strike event conditions pass, a payout is triggered for the identified farmer. To demonstrate the usefulness of the system, a simpler approach for triggering payouts using strike event condition checking is used, so as to navigate the potentially time-consuming and complex process of building indices. The component of constructing indices is left for further exploration. Figure 16 provides an illustration of a payout being triggered when the strike conditions attached to an insurance policy are met.

[29] https://bima.onrender.com/weather

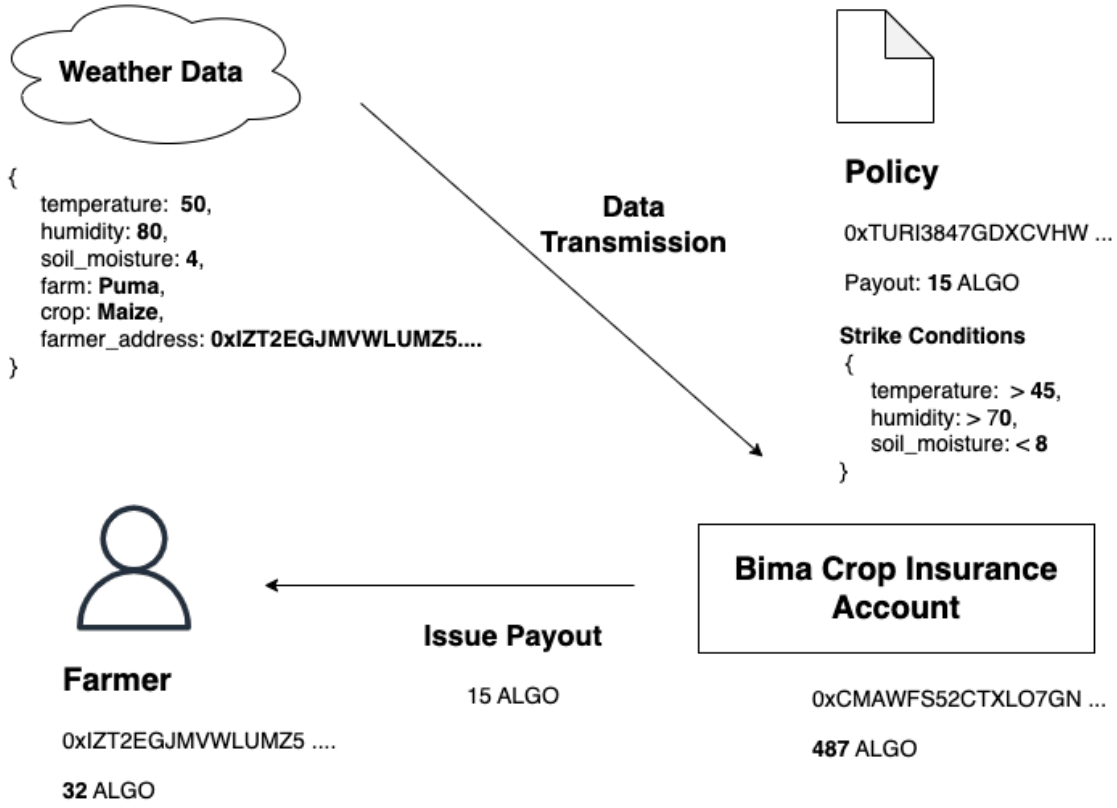

Figure 16: Illustration of a payout being triggered when the strike event conditions attached to an insurance policy are met. Weather data that is sent from sensors installed on a farmer's farm is checked against strike event conditions stipulated in the farmer's insurance policy. If strike event conditions pass, a payout is triggered for the farmer.

4.6 Platform Screens

In this section, I outline the user interface screens that make up the weather index crop insurance platform. The article on principles of user interface design Porter (2023) outlines some of the guidelines I use in creating the user interface which include one primary action per screen, maintaining clarity and consistency on each screen.

4.6.1 Farmer Portal Login Page

Farmers log in with their 25-word passphrase. If they do not have an account, they can click on the *Create Account* button and an account will be created for them. Upon account creation, a farmer will receive a passphrase to their account that they can use to log back on the platform.

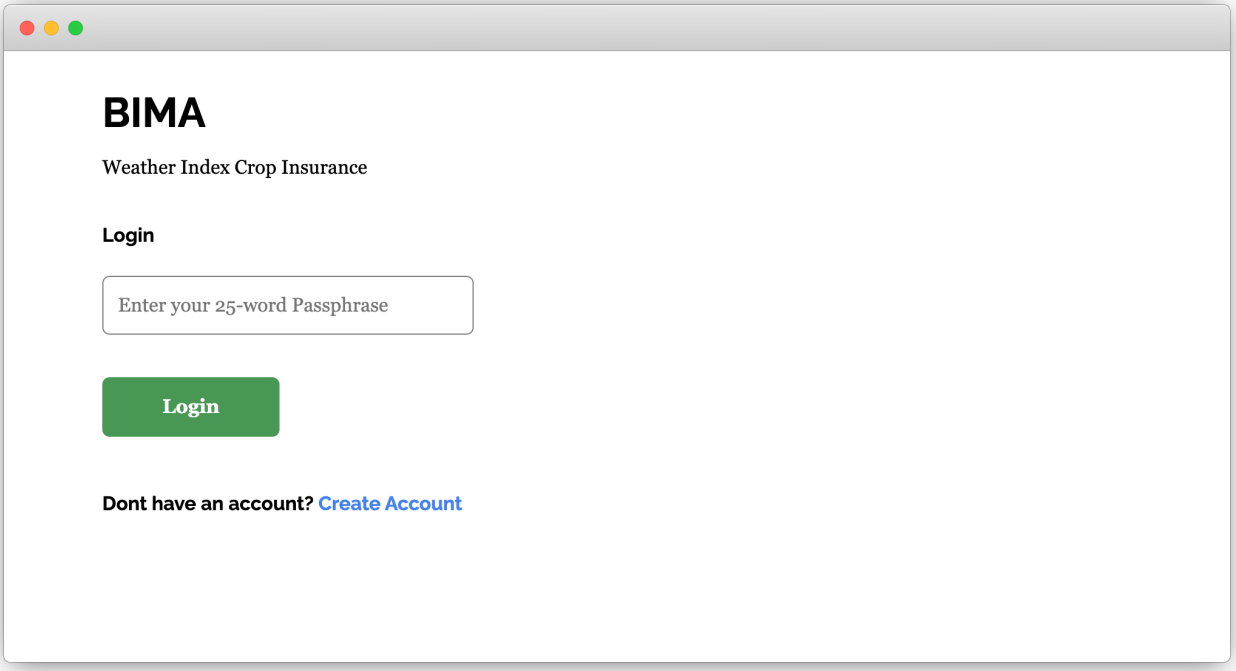

Figure 17: Farmer portal login screen. To log in a farmer enters their 25-word mnemonic and clicks the Login button. To sign up for the service a farmer clicks the Create Account button which creates an Algorand blockchain account for them and displays their mnemonic which they can use to log on to the platform.

4.6.2 Farmer Account Details Page

Upon successful login, a farmer can view their account details on the account details page. On the page, a farmer is able to see:

- Their account wallet address and account's ALGO balance.
- The policies they have joined. By clicking on the *Policy on chain* button, a farmer is able to view the details of the policy stored on the blockchain.
- They are able to pay their premiums for the current month, by clicking on the *Pay Premium* button next to each joined policy and the transaction will be stored on-chain.
- A farmer is able to cancel a policy by clicking the *Cancel* button.

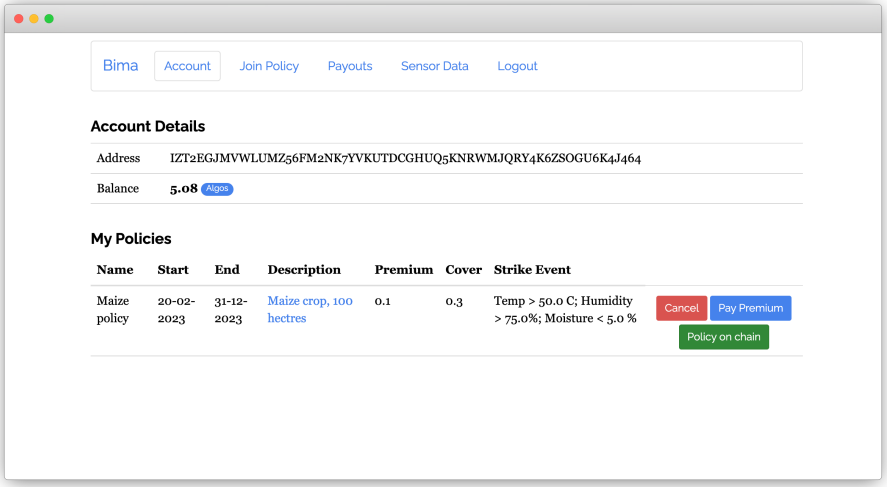

Figure 18: Farmer portal account details page. On this page, a farmer views their account details including their wallet address, ALGO balance and a list of the policies they have joined. They can pay their premium by clicking the Pay Premium button. To view the policy details logged on the Algorand blockchain a farmer clicks the Policy on chain button. To cancel a policy a farmer clicks the Cancel button on the corresponding policy.

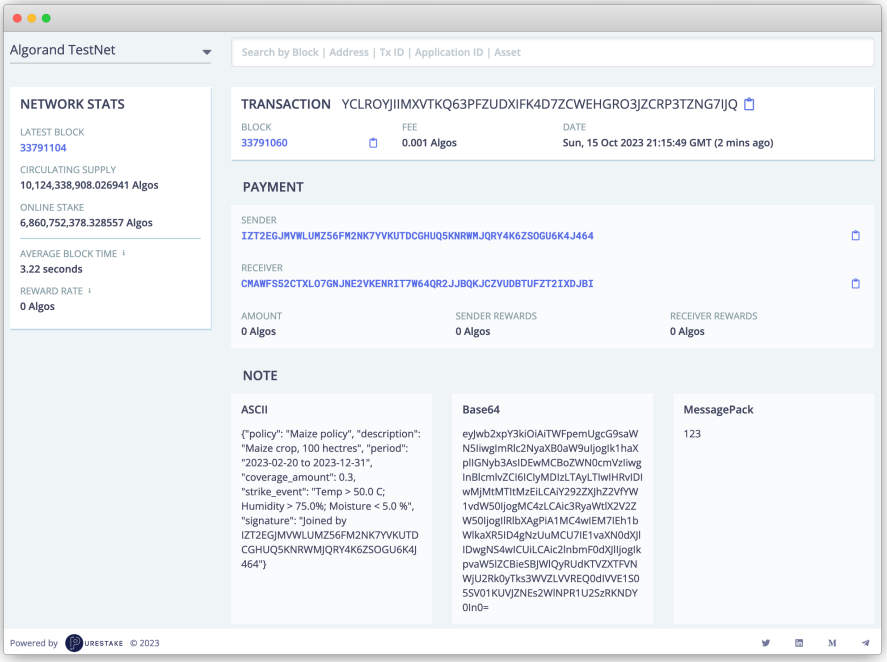

Figure 19: A farmer's joined policy details stored on the Algorand blockchain. When a farmer joins an insurance policy on the platform the details of the farmer's new policy are logged on the Algorand blockchain as a transaction with the policy details included in the transaction note.

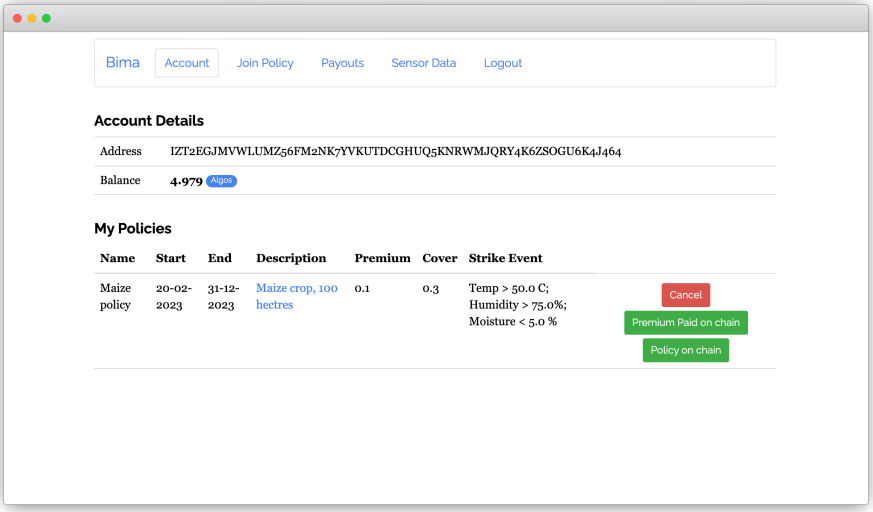

Figure 20: Farmer portal account details page showing a policy activated when the premium has been paid. To view the details of the paid premium on the Algorand blockchain a farmer clicks the Premium Paid on chain button.

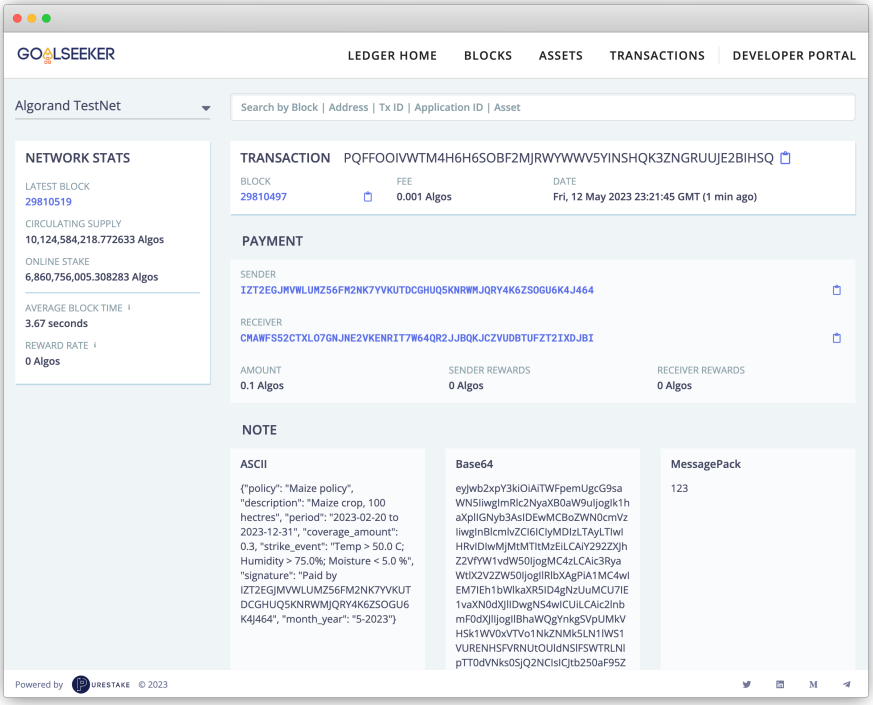

Figure 21: Details of the paid premium stored on the Algorand blockchain. When a farmer pays a premium the details of the paid premium and the corresponding policy are logged on the Algorand blockchain as a transaction with the details included in the transaction note.

4.6.3 Crop Insurance Policy Details Page

On the policy page, a farmer can join an insurance policy, choosing from the available listed policies by clicking on the *Join* button next to the desired policy. On this page, a farmer is also able to cancel joined policies by clicking on the *Cancel* button next to a joined policy.

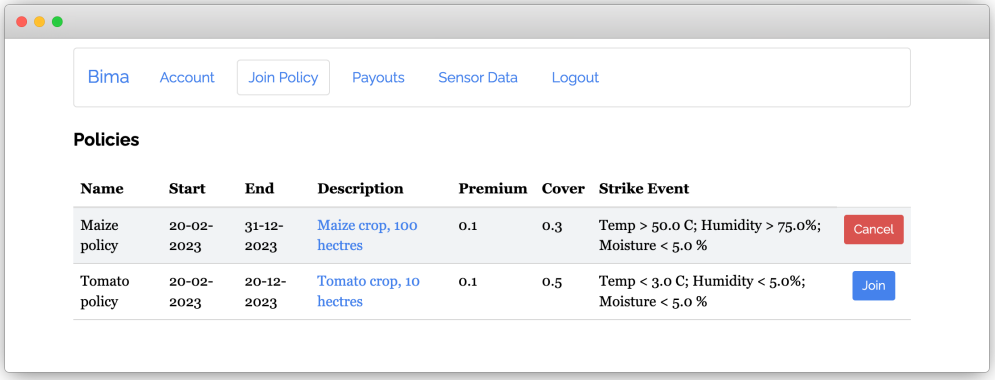

Figure 22: Farmer portal insurance policy details page lists all the available insurance policies on the platform. On the policy page, a farmer can join a policy, choosing from the available listed policies by clicking on the Join button or cancel a policy by clicking the Cancel button.

4.6.4 Insurance Payouts Page

Insurance payouts are auto-processed when a strike event attached to the farmer's policy is triggered. On the payouts page, a farmer can see a list of all the payouts they were issued and when each payout was processed. By clicking on the *View on chain* button a farmer is able to verify the payout details stored on-chain.

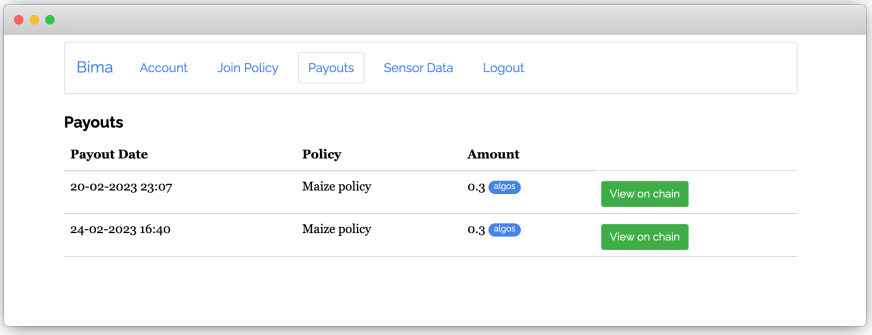

Figure 23: Farmer portal payout page lists all the payouts that have been issued to a farmer, the payout date, the corresponding policy and the payout amount. A farmer is able to view the payout transactions on the Algorand blockchain by clicking on the View on chain button.

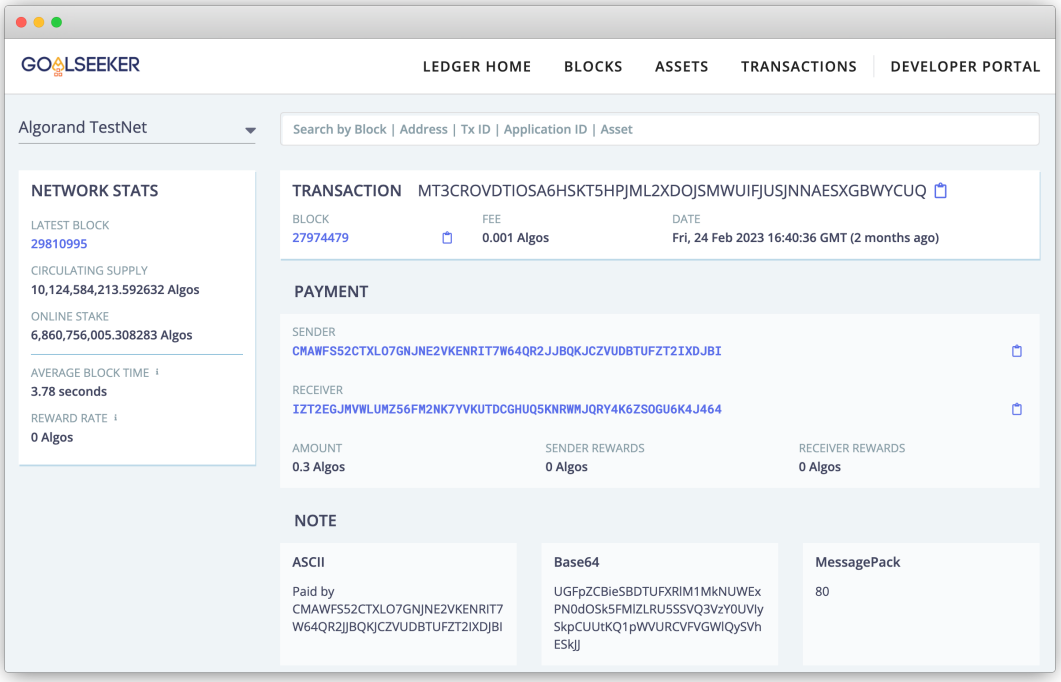

Figure 24: Payout transaction details stored on the Algorand blockchain. When a farmer is issued a payout, the details of the payout amount and who made the payment are logged as a transaction on the Algorand blockchain.

4.6.5 Sensor Data Page

On the sensor data page, a farmer will see the weather data collected by the sensors installed on their farm:

- As a graph with the most recent data displayed first.
- As a table of the weather data records.
- A farmer is able to view the data stored on the blockchain by clicking on the *View on chain* button.

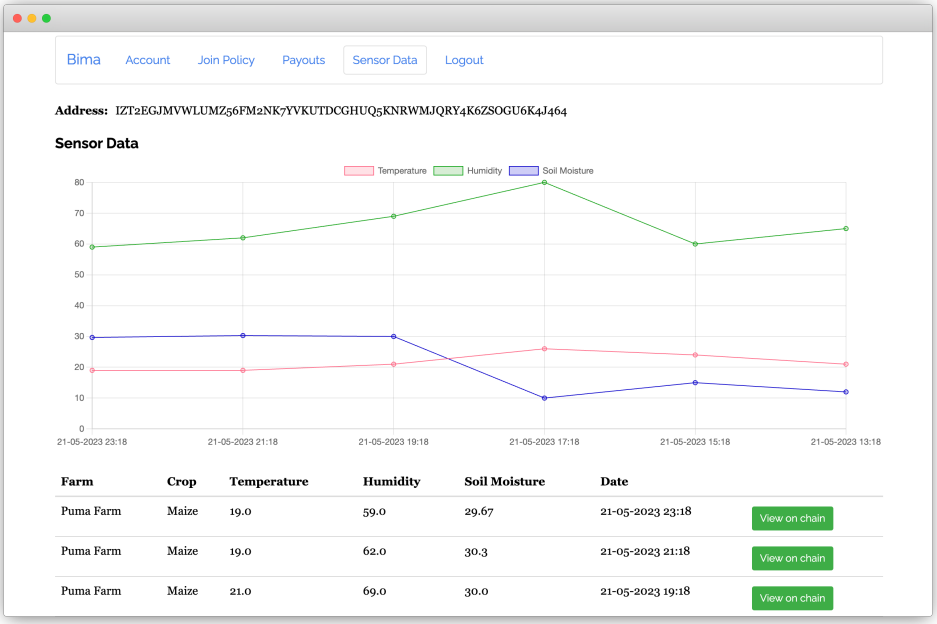

Figure 25: Farmer portal sensor data page lists sensor data readings from the farmer's farm and also presents the readings as a graph. The readings are logged to the Algorand blockchain and a farmer can view the logged data by clicking the View on chain button.

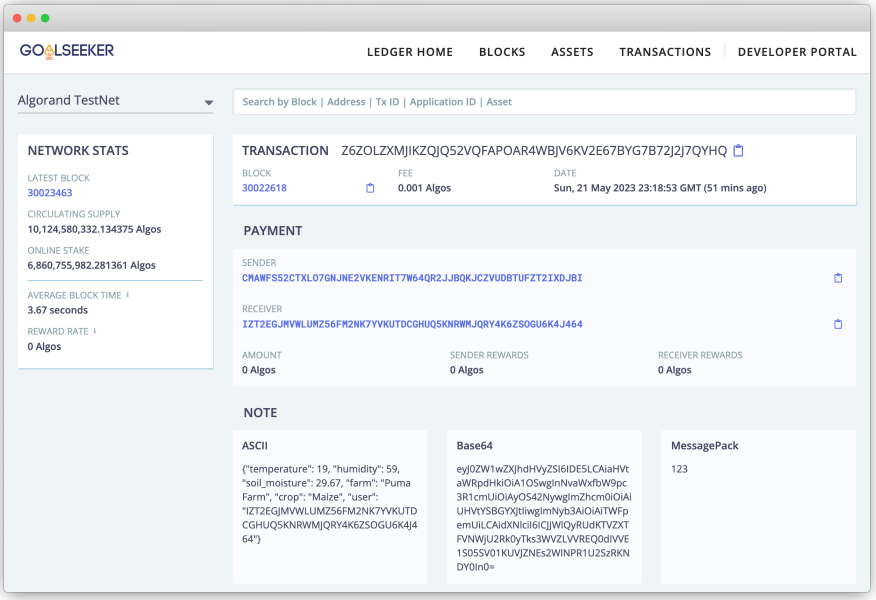

Figure 26: Sensor data details stored on-chain. Sensor data received on the platform is logged as a transaction on the Algorand blockchain and the sensor data is included in the transaction note.

4.6.6 Administration Portal

System administrators have a separate portal where they are able to log in on the platform to perform administrative tasks such as adding insurance policy details. Administrators have create, read, update and delete functionality for the models in the system including, users and their roles, farms, policies and strike events.

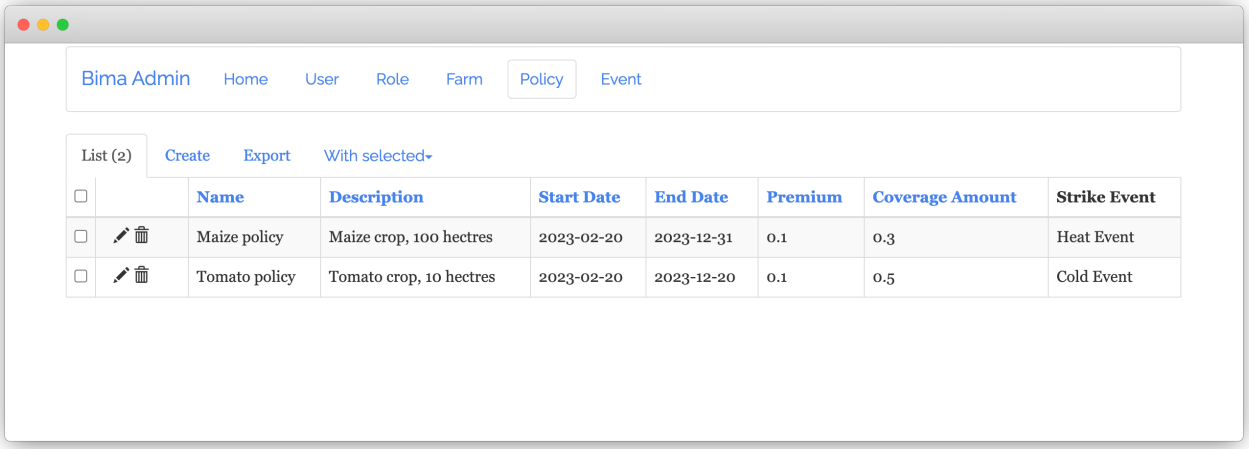

Figure 27: Administration portal interface and functionality. On this portal system administrators are able to perform administrative tasks such as adding insurance policy information and updating records.

5 Evaluation

Testing is a critical part of a software development life cycle that ensures the reliability and performance of software applications. The aim of software testing is to detect defects at the earliest possible stage (Gojare et al., 2015). Through testing, software defects in the platform are identified and fixed, and an evaluation of the platform's performance under different conditions is done. Valuable insights into the platform's overall effectiveness are gained. In developing the prototype for the weather index crop insurance platform, I test the platform to ensure that it simulates closely the needs of its users and functions as intended from the requirements specification. In this testing and evaluation section, I discuss the test methodology and results, I observe from the platform.

5.1 System Testing

System testing evaluates the platform to ensure that it meets the specified requirements and works as intended (Baresi and Pezzè, 2006). The platform is tested with parameters that closely resemble the end-user experience. Various aspects of the system including its functionality, performance, security, and reliability are tested.

I use the following criteria to evaluate the success of the system testing:

- All functional and non-functional requirements are met. See Table 3.
- No critical defects are found that impact the application's performance or usability.
- The application's performance meets or exceeds the specified requirements.
- The application's security features are validated and any vulnerabilities are identified and addressed.

System testing helps to ensure the overall reliability and effectiveness of the weather index crop insurance platform. Through system testing, I note areas of improvement, such as transitioning from using ALGO as a form of payment on the platform to using stablecoins (Fiedler and Ante, 2023) such as USDC[30], which may be easier for farmers who are more familiar with fiat money, to understand and transact with. Table 4 provides a list of the test cases that form the basis of system testing on the platform. The test cases I conduct on the platform pass, which gives confidence in the stability and reliability of the platform.

5.2 Regression Testing

Regression testing is a type of software testing that is performed to ensure that changes or modifications made to a software application have not introduced new bugs or caused unintended side effects

[30] https://developers.circle.com/developer/docs/usdc-on-testnet

	Requirement	Comment
FR1	The platform should allow for weather data management - weather data capture using IoT sensors, storage of the weather data on-chain and publishing of the data via an API. The data should be relevant to the farmer's field location and crops.	Satisfied
FR2	Digital insurance policy management - the process of insurance policy creation and farmers joining policies should be managed via the platform.	Satisfied
FR3	The platform should have an automated payout system on-chain that is secure and reliable. Insurance payouts should be triggered by strike conditions configured per insurance policy.	Satisfied
FR4	Users should be able to easily make premium payments and receive payouts.	Satisfied
FR5	The system should have access control for sensitive client details.	Satisfied
NFR1	The platform should be user-friendly and not too technically complicated.	Satisfied
NFR2	Users should be able to access relevant information and features with minimal effort.	Satisfied
NFR3	The proposed system should have a reliable storage system where all processing information can be stored.	Satisfied
NFR4	The platform should be responsive and accessible from various devices, including desktop and mobile.	Satisfied
NFR5	There should be minimum latency in information propagation in the system.	Satisfied
NFR6	The proposed platform should process data accurately, quickly and automatically.	Satisfied
NFR7	Processes like calculations should be done efficiently without errors.	Satisfied

Table 3: Functional (FR*) and non-functional (NFR*) requirements with an indication of whether each is satisfied. All requirements are met.

on previously tested functionality (Yoo and Harman, 2012). It verifies that existing features and functionalities of the application continue to work as intended after any changes or updates have been made. Regression testing, combined with automation testing, plays a vital role in ensuring the reliability and accuracy of the platform. By automating the execution of test cases, I efficiently and thoroughly test various functionality and scenarios on the platform. I leverage automation testing using the Selenium Webdriver[31] to validate the platform's user interface and user experience. I create

[31] https://www.selenium.dev/documentation/webdriver/

Test Category	Test Case Objective	Description	Expected Result	Result (Pass/ Fail)
Data security	Farmer account creation	Creation of an Algorand wallet for a farmer	Create account button creates an account for the farmer and shows them their mnemonic to use when logging in	Pass
Policy administration	Insurance policy listing	Visiting the policy page should show a farmer the available policies to join	Farmer can view available policies	Pass
Policy administration	Joining policy	Farmer should be able to join a policy and the details stored on the blockchain	Farmer can join a policy if they have funds to pay for the transaction fee	Pass
Insurance payments	Premium payments	Farmers should be able to pay premiums to activate their policy	Premium payment transactions succeed and activates policy	Pass
Insurance payments	Payouts	Payouts should be triggered when conditions attached to a strike event on a policy are surpassed	Data sent from sensors which is outside the range of the conditions on the strike event should trigger a payout	Pass
Data streaming	Weather data	Sensor data must be sent to the platform via the API	Temperature, humidity and soil moisture data sent from the sensors is stored on and off-chain	Pass

Table 4: Test cases I use for system testing. The test cases span across the insurance policy administration, weather data collection, insurance premium payments and payouts activities. All of the test cases pass.

scripts[32] that simulate user interactions, such as logging in on the platform, performing actions like joining a policy, and simulating adverse weather conditions to trigger payouts on the platform. This testing approach helps identify any discrepancies or inconsistencies in the system. Another crucial aspect of automation testing is its application in regression testing. As I add new features to the platform, it is essential to ensure that existing functionality remains unaffected. Automation allows me to execute a suite of regression tests that cover various use cases, ensuring the overall stability of the platform. This way, I can confidently validate the platform's performance and maintain its functionality as I introduce enhancements.

[32] https://github.com/rideam/bima/tree/main/tests

5.3 Communication Rounds

The weather index crop insurance platform offers a streamlined approach with fewer communication rounds, leveraging predetermined triggers for faster payouts and policy enrollment. In contrast to traditional insurance schemes, which involve multiple rounds of communication between users, insurance companies, and stakeholders, the platform provides a quicker and more convenient process. The benefits for users include simplified enrollment, reduced administrative burden, and faster payout processing. These benefits are further explained below:

i) A simplified enrollment process is more convenient and easier for individuals seeking insurance coverage to understand their options, choose the appropriate coverage and complete the enrollment process with minimal hassle.
ii) Fewer communication rounds reduce the administrative burden on both the insurer and the user seeking insurance, making the platform cheaper and more convenient.
iii) Quicker payout processing ensures policyholders receive the compensation they need in a timely manner.

Table 5 compares communication rounds between the Bima weather index crop insurance platform and traditional insurance schemes. The Bima platform has fewer communication rounds.

Operation	**Scheme**	**Rounds**	**Comment**
Insurance Policy Enrollment	Bima Platform	• Enroll by clicking Join • Activate with premium payment	• 2 rounds • Farmer and platform
	Traditional Insurance Schemes	• Application submission • Underwriting and risk assessment • Policy offer and negotiation • Policy Acceptance • Premium Payment	• Minimum 5 rounds: • Farmer and Insurer
Claim Processing	Bima Platform	• Weather data triggers strike event • Payout auto-processed	• 2 rounds: • Microcontroller and platform • Farmer and platform
	Traditional Insurance Schemes	• Claim filing and supporting documentation submission • Claim verification • Claim settlement	• Minimum 3 rounds: • Farmer and Insurer • Insurer and Agents

Table 5: A comparison of communication rounds for the weather index crop insurance platform and traditional insurance schemes. The Bima weather index crop insurance platform requires fewer communication rounds between users, insurance providers, and stakeholders, benefitting users with a simple enrollment process, reduced administrative burden, and faster payout processing.

5.4 Load Testing

Load testing is a type of performance testing that assesses how the platform responds under normal and excessive demand (Shrivastava and Prapulla, 2020). Load testing is done to ensure that the platform can handle the expected traffic load without any significant loss of performance. I use simulated traffic on the platform to measure its capacity and capabilities. I carry out load testing on the platform by sending requests to endpoints that simulate users performing different actions on the platform: i) the action of joining an insurance policy[33] and the transaction being stored on-chain; ii) microcontrollers sending data and payouts being triggered[34]; and iii) general platform access through accessing the login page[35]. Load testing helps to determine the concurrent user capacity, analyse platform response time, resource utilization, and identify potential bottlenecks that can degrade the performance of the platform. I use Locust[36], a package for load testing to write and execute the scripts to load test the platform. I compare the performance of the platform with and without load balancing. Load balancing is necessary to achieve efficient resource utilization, improve system performance, ensure high availability and reliability, and provide a better user experience. By evenly distributing workloads across multiple resources, load balancing reduces response times and improves the overall performance of the platform, enabling scalability, flexibility, and efficient resource management. Load testing results show that the response time of the platform is significantly improved by applying load balancing with NGINX[37]. See Appendix B for the load-testing test cases and graphical results.

5.4.1 Insurance Policy Joining

Table 6 shows the load testing results from testing the endpoint that simulates users joining an insurance policy on the platform and the corresponding concurrent user count in each test without load balancing. Figure 28 shows the load testing results graphically. Exploring the relationship between these response time metrics and the corresponding user counts shows that median and average response time increases as the user count increases. This result suggests a positive correlation between user count and the time it takes for the system to process insurance policy joining requests without load balancing. The maximum response time also shows a similar trend increasing as the number of users increases. Table 7 shows the load testing results from testing the endpoint that simulates users joining an insurance policy on the platform and the corresponding concurrent user count in each test when the user requests are load balanced with NGINX. Figure 29 shows the load testing results with load balancing graphically. Comparing the results in Table 6 and Table 7 we see

[33] https://bima.onrender.com/joinpolicytest
[34] https://bima.onrender.com/payouttest
[35] https://bima.onrender.com/login
[36] https://locust.io/
[37] https://www.nginx.com/

that load balancing significantly improves the response time of the platform. The median and average response times significantly improve and the maximum response time of the platform is capped at 60 seconds as the number of concurrent users increases beyond 10.

	Response Time Without Load Balancing			
User Count	**Median Response Time (s)**	**Average Response Time (s)**	**Min Response Time (s)**	**Max Response Time (s)**
1	6.16	6.16	6.16	6.16
3	29.00	25.52	7.26	36.46
10	52.00	51.53	21.93	73.53
20	110.00	102.50	22.49	146.61

Table 6: Load testing results without load balancing, for joining an insurance policy on the platform as the number of concurrent users accessing the platform increases showing response time and user count. The load test is performed against the joinpolicytest endpoint which simulates a user joining a policy and transaction details being stored on-chain.

	Response Time With Load Balancing			
User Count	**Median Response Time (s)**	**Average Response Time (s)**	**Min Response Time (s)**	**Max Response Time (s)**
1	6.60	7.05	6.36	8.23
3	13.14	16.42	13.14	19.70
10	33.00	37.30	8.88	60.03
20	29.00	35.69	9.51	60.02

Table 7: Load testing results with load balancing for joining an insurance policy on the platform as the number of concurrent users accessing the platform increases showing response time and user count. The load test is performed against the joinpolicytest endpoint which simulates a user joining a policy and transaction details being stored on-chain. These results show that load balancing significantly improves the response time of the platform as compared to the results without load balancing.

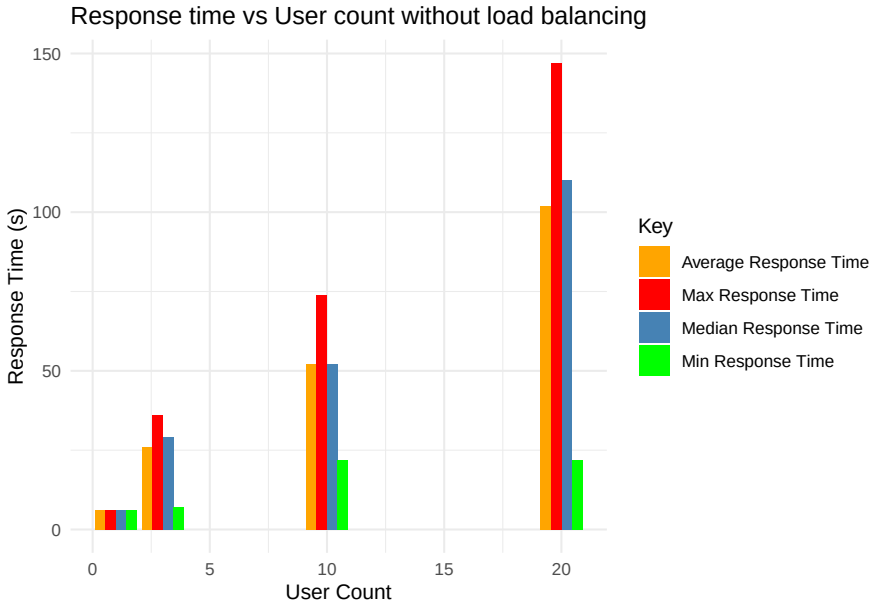

Figure 28: Response time of the platform without load balancing, as the number of users and transactions increases. Max, median and average response times all increase as the number of users increases but minimum response time shows a small variation showing that the system can still achieve a minimum response time even with higher load.

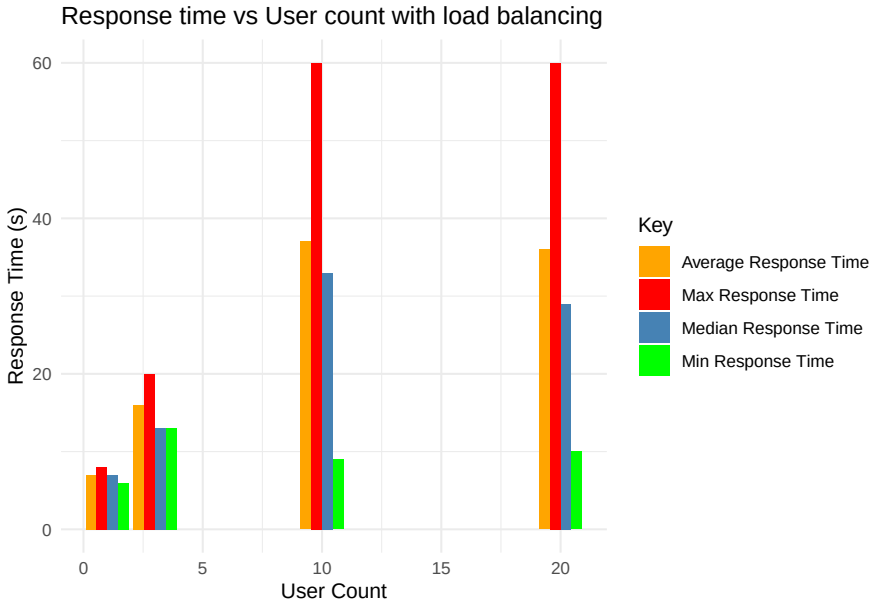

Figure 29: Response time of the platform with load balancing, as the number of users and transactions increases. Response time is significantly improved with load balancing. Max response time is capped at 60 seconds as the number of user increases beyond 10.

5.4.2 Insurance Payout Processing

Similarly, payout processing load testing is done to investigate how much time it takes for insurance payouts to be processed on the platform as load increases. Table 8 shows the platform payout processing load testing results and the corresponding user count in each test without load balancing. Figure 30 shows the payout processing load testing results graphically. The results also show a positive correlation between the system's average, median and maximum response time as load increases without load balancing. The graph reveals a relatively consistent minimum response time across different user counts implying that even with a higher load, the system can still achieve a minimum payout processing time with a small variation range. Table 9 shows the platform payout processing load testing results and the corresponding user count in each test with load balancing. Figure 31 shows the payout processing load testing results graphically. Comparing the results in Table 8 and Table 9 we see that load balancing significantly improves the payout processing response time of the platform. The median and average response times significantly improve and the maximum response time of the platform is capped at 60 seconds as the number of concurrent users increases beyond 10.

	Response Time Without Load Balancing			
User Count	**Median Response Time (s)**	**Average Response Time (s)**	**Min Response Time (s)**	**Max Response Time (s)**
1	15.7	15.70	15.70	15.70
3	31.0	31.80	16.08	44.03
10	59.0	57.38	14.58	88.22
20	118.0	117.25	15.17	161.29

Table 8: Insurance payouts processing load testing results without load balancing, showing response time and concurrent user count. The load test is performed against the payouttest endpoint which simulates data being sent to the platform from microcontrollers and payout transactions being triggered on the platform if strike conditions are met.

User Count	Response Time With Load Balancing			
	Median Response Time (s)	Average Response Time (s)	Min Response Time (s)	Max Response Time (s)
1	13.01	13.01	13.01	13.01
3	13.05	14.46	13.05	18.55
10	39.00	40.38	15.32	60.01
20	46.00	43.63	16.03	60.05

Table 9: Insurance payouts processing load testing results with load balancing, showing response time and concurrent user count. The load test is performed against the payouttest endpoint which simulates data being sent to the platform from microcontrollers and payout transactions being triggered on the platform if strike conditions are met. These results show that load balancing significantly improves the response time of the platform as compared to the results without load balancing.

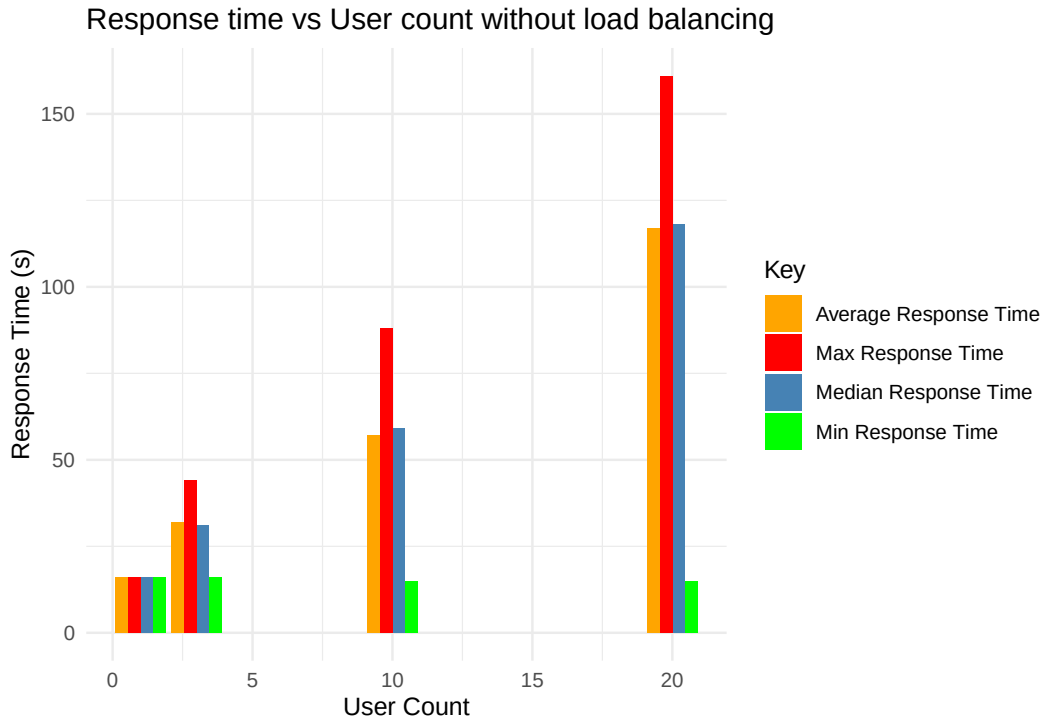

Figure 30: Response time of the platform without load balancing as the number of transactions increases. Max, median and average response times all increase as the number of transactions increases but minimum response time remains constant showing that the system can still achieve a minimum payout processing response time even with higher load.

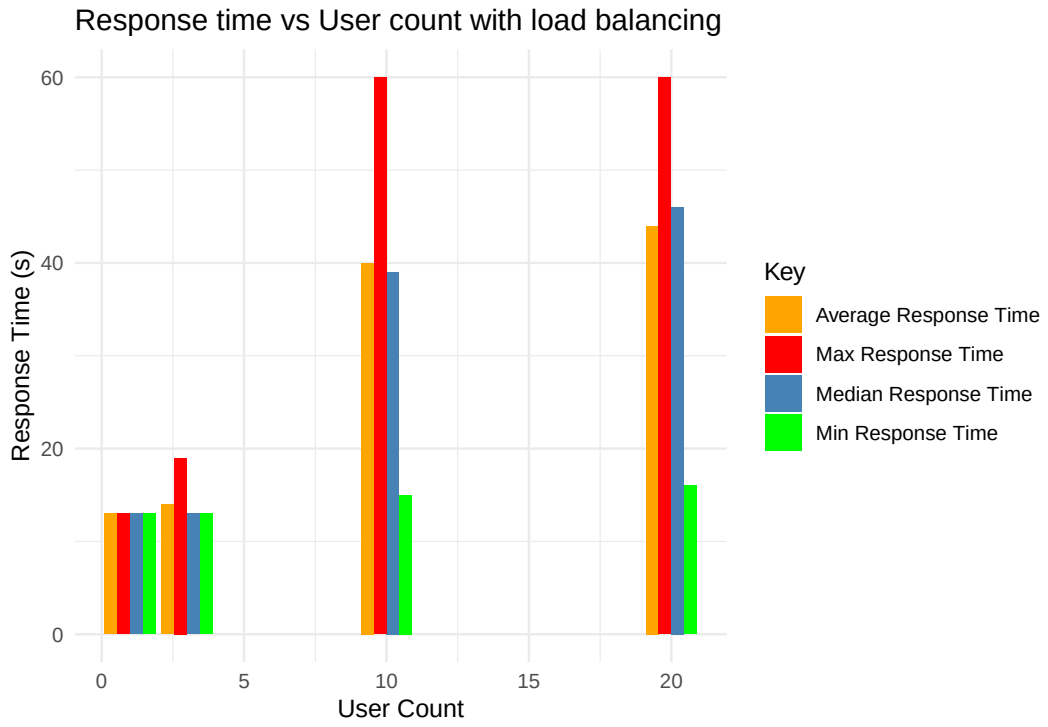

Figure 31: Response time of the platform with load balancing as the number of transactions increases. Response time is significantly improved with load balancing. Max response time is capped at 60 seconds as the number of users increases beyond 10.

5.4.3 General Platform Access

General platform access load testing without load balancing shows that the response time on the platform increases as the number of users increases. Table 10 shows the platform's general access load testing results and the corresponding concurrent user count in each test without load balancing. Figure 32 shows the general platform access load testing results graphically. The results show a positive correlation for all the metrics when measuring the general response time of the platform as load increases without load balancing. Table 11 shows the platform's general access load testing results and the corresponding concurrent user count in each test with load balancing. Figure 33 shows the general platform access load testing results graphically. Comparing the results in Table 10 and Table 11 we see that load balancing significantly improves the response time of the platform. The max, median and average response times significantly improve.

User Count	Response Time Without Load Balancing			
	Median Response Time (s)	Average Response Time (s)	Min Response Time (s)	Max Response Time (s)
1	1.1	1.21	1.08	1.60
3	7.3	7.61	1.24	14.65
10	51.0	54.40	22.11	88.23
20	110.0	110.91	51.73	161.29

Table 10: Platform general access load testing results without load balancing, showing response time and user count. The load test is performed against the login endpoint which is the landing page when accessing the platform.

User Count	Response Time With Load Balancing			
	Median Response Time (s)	Average Response Time (s)	Min Response Time (s)	Max Response Time (s)
1	0.04	0.08	0.03	0.16
3	6.60	7.38	0.01	18.59
10	26.00	25.12	0.10	45.88
20	33.00	29.65	0.09	42.39

Table 11: Platform load testing results with load balancing showing response time and user count. The load test is performed against the login endpoint which is the landing page when accessing the platform. These results show that load balancing significantly improves the response time of the platform as compared to the results without load balancing.

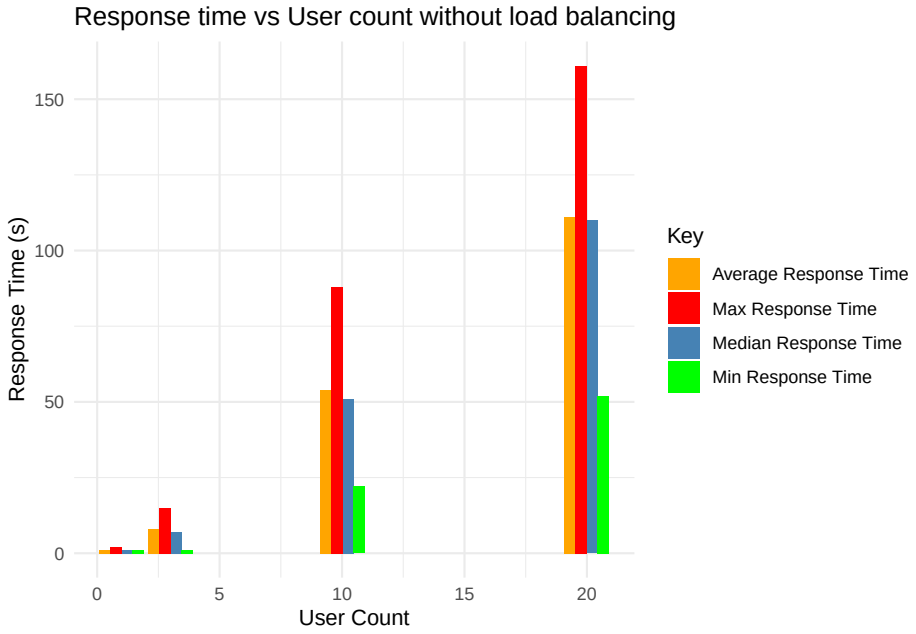

Figure 32: Response time of the platform without load balancing as the number of users increases. Response time increases as the number of users accessing the platform increases.

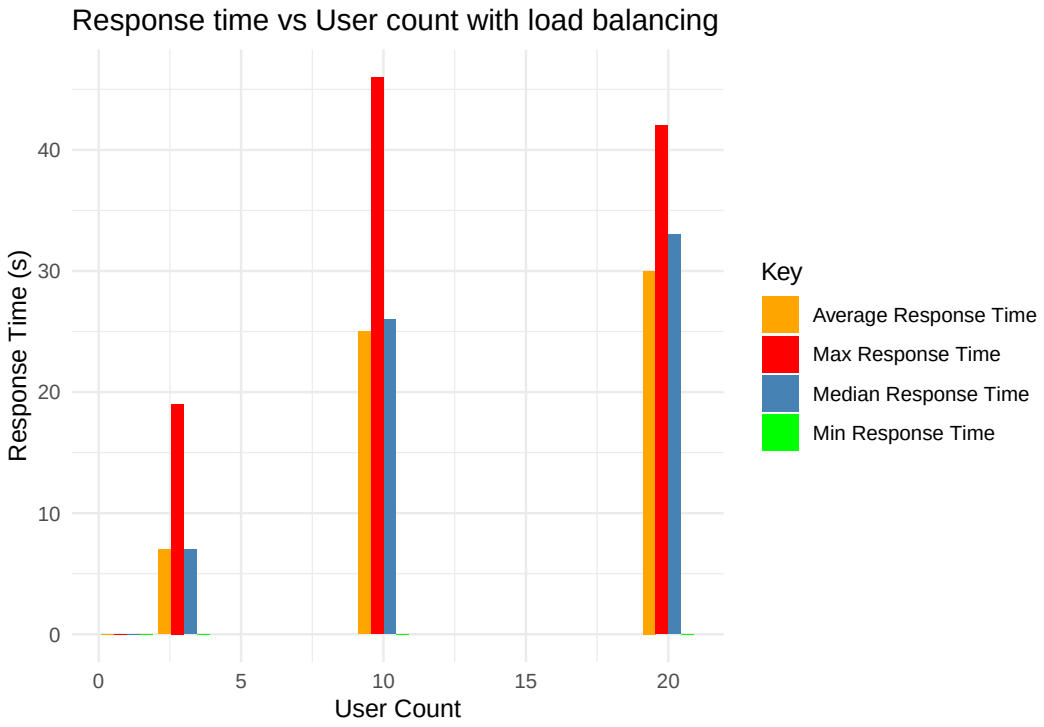

Figure 33: Response time of the platform with load balancing as the number of users increases. Response time is significantly improved with load balancing.

6 Discussion

The weather index crop insurance platform developed in this research project demonstrates significant potential to improve agricultural resilience and promote food security in the face of climate change and weather-related risks. By implementing a range of key features, including weather data collection from sensors, user authentication, digital policy management, instant auto-triggered payouts, and transactions stored on-chain, the platform creates a more accessible, reliable, and transparent system for smallholder farmers to manage weather-related risks and protect their livelihoods. One of the main strengths of the platform is its ability to leverage advanced technologies, such as blockchain and in-field sensors, to create a more efficient and effective system for weather data collection, verification, and processing. By storing transactions on the blockchain, the platform has increased transparency and accountability, reducing the potential for fraud and ensuring that payouts are delivered quickly. Similarly, by collecting weather data from sensors, the platform has improved the density of collected data which can be used to build indices, making the payout process more reliable and relevant for farmers as individualised indices can be formulated from the collected weather data to reduce basis risk. Another key strength of the platform is its user-friendly design and functionality which ensures that non-technical users can easily understand and use the platform. By implementing features such as user authentication and policy joining, the platform has made it easier for farmers to access and manage insurance policies, reducing barriers to entry and increasing the likelihood of index insurance uptake. Similarly, by providing timely and accurate payouts, the platform demonstrates its value to smallholder farmers as they will receive their payouts timeously and builds trust in the system as farmers can verify the transactions stored on the blockchain.

However, there are also several areas for improvement and further development. For example, while the use of sensors for weather data collection is a significant step forward, there is still room for improvement in terms of data quality, reliability, and coverage and actually building the algorithms to calculate the indices to be used to determine payouts. By continuing to invest in research and development, and working closely with farmers and other stakeholders, the platform has the potential to make a significant contribution towards sustainable and equitable agricultural development.

6.1 Future Work

Looking ahead, several areas of future work will further enhance the weather index crop insurance platform and its impact on promoting agricultural resilience and improving food security:

- The is scope to enhance the platform by actually developing the algorithm that builds weather-based indices to be used to determine payouts.
- The is also scope to use additional sensors to collect more data that can be used to build weather-based indices. For example, incorporating sensors such as the tipping bucket rainfall gauge to measure the amount of rainfall so as to increase the number of available insurance products.
- Using stablecoins like USDC as a payment method on the platform can help farmers who are more familiar with fiat money to better understand the transactions, as compared to the current implementation that uses the ALGO cryptocurrency. Similarly, supporting retail central bank digital currency (CBDC) is a natural extension.
- The platform can be integrated with other agricultural services and technologies, such as mobile money or e-commerce platforms, to create a more seamless and integrated user experience. This could also facilitate the delivery of payments and other services to farmers from a single platform.
- Making the platform accessible via low-tech interfaces e.g. Unstructured Supplementary Service Data (USSD), Short Message Service (SMS), WhatsApp etc. to better support smallholder farmers who may be located in remote regions and have limited access to smartphones or connectivity.
- Making use of Algorand smart contracts for data storage and automated payments, instead of using the Python Algorand SDK accounts, and zero-value transactions for data exchange.
- Using device identity management/authentication to prevent unauthorised microcontrollers from sending weather data from sensors to the weather data API and writing to the blockchain.
- Further exploration of additional methods to secure and maintain the privacy of sensitive data beyond password authentication, to address the other various aspects of data protection that were not covered in this thesis.
- The platform should be piloted in a real-life scenario with smallholder farmers to assess its practical utility, beyond the technical and theoretical usefulness that is demonstrated in this thesis.

7 Conclusion

In conclusion, the implementation of the weather index crop insurance platform is a success, with several key features that enhance the smallholder farmer experience with insurance provision. With user authentication, farmers can be confident that their personal data and policy information is secure and protected. The ability to join low-cost policies on the platform makes it easier for farmers to access insurance and manage their risks, while auto-triggered payouts provide timeous, hassle-free financial support in the face of weather-related crop losses. The platform streamlines the insurance process by eliminating lengthy claims processing with the use of auto-triggered payouts, that are triggered when strike conditions in a policy are met. Additionally, it digitizes policy management, reducing the dependence on paper. The collection of weather data is a critical aspect of the platform, providing accurate and reliable information that is used to determine payouts. By storing transactions on the Algorand blockchain, the platform is able to ensure transparency and accountability, enabling farmers and other stakeholders to track transactions and verify the accuracy of payouts.

Overall, the weather index crop insurance platform developed in this research project demonstrates the potential to promote agricultural resilience and improve food security. This is true, particularly in emerging economies where farmers are vulnerable to weather-related risks, by making crop insurance cheaper, more accessible and claims processing faster, more secure, and more efficient.

References

Afriyie-Kraft, L., Zabel, A., and Damnyag, L. (2020). Index-based weather insurance for perennial crops: A case study on insurance supply and demand for cocoa farmers in ghana. *World Development Perspectives*, 20:1–6. Available at https://www.sciencedirect.com/science/article/pii/S2452292920300576.

Aggour, K. S., Bonissone, P. P., Cheetham, W. E., and Messmer, R. P. (2006). Automating the underwriting of insurance applications. *AI Magazine*, 27(3):36. Available at https://doi.org/10.1609/aimag.v27i3.1891.

Algorand (2023). What is a blockchain? https://developer.algorand.org/docs/get-started/basics/what_is_blockchain. [Accessed: February 20, 2023].

Baresi, L. and Pezzè, M. (2006). An introduction to software testing. *Electronic Notes in Theoretical Computer Science*, 148(1):1–23. Proceedings of the School of SegraVis Research Training Network on Foundations of Visual Modelling Techniques (FoVMT 2004). Available at https://www.sciencedirect.com/science/article/pii/S1571066106000442.

Bhusal, C. S. (2021). Blockchain technology in agriculture: A case study of blockchain start-up companies. *International Journal of Computer Science and Information Technology (IJCSIT)*, 13:39–41. Available at SSRN:https://ssrn.com/abstract=3960631.

Burke, M., de Janvry, A., and Quintero, J. (2010). Providing index based agricultural insurance to smallholders: Recent progress and future promise. California: CEGA, University of California Berkeley:1–4. Available at https://web.worldbank.org/archive/website01589/WEB/IMAGES/PARAL-18.PDF.

Cesarini, L., Figueiredo, R., Monteleone, B., and Martina, M. L. V. (2021). The potential of machine learning for weather index insurance. *Natural Hazards and Earth System Sciences*, 21(8):2379–2405. Available at https://nhess.copernicus.org/articles/21/2379/2021/.

Chen, J. and Micali, S. (2019). Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183. In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I. Available at https://doi.org/10.1016/j.tcs.2019.02.001.

Circuitschools (2023). Interface capacitive soil moisture sensor v1.2 with arduino, lcd and oled. https://www.circuitschools.com/interface-capacitive-soil-moisture-sensor-v1-2-with-arduino-lcd-and-oled/. [Accessed: January 4, 2023].

Collier, B., Skees, J., and Barnett, B. (2009). Weather index insurance and climate change: Opportunities

and challenges in lower income countries. *The Geneva Papers on Risk and Insurance - Issues and Practice*, 34(3):pp 401–424. Available at https://doi.org/10.1057/gpp.2009.11.

Components101 (2023). Dht11–temperature and humidity sensor. https://components101.com/sensors/dht11-temperature-sensor. [Accessed: January 4, 2023].

Desiree, D. and Chinwe, I. S. (2020). The role of blockchain in documenting land users' rights: The canonical case of farmers in the vernacular land market. *Frontiers in Blockchain*, 3:1–8. Available at https://doi.org/10.3389/fbloc.2020.00019.

Espressif (2023). Esp32. https://www.espressif.com/en/products/socs/esp32. [Accessed: January 4, 2023].

Fiedler, I. and Ante, L. (2023). *Stablecoins, The Emerald Handbook on Cryptoassets: Investment Opportunities and Challenges*, pages 93–105. Emerald Publishing Limited. Available at https://doi.org/10.1108/978-1-80455-320-620221007.

Ghosh, R. K., Gupta, S., Singh, V., and Ward, P. S. (2020). Demand for crop insurance in developing countries: New evidence from india. *Journal of Agricultural Economics*, 72(1):pp 293–320. Available at https://doi.org/10.1111/1477-9552.12403.

Gine, X. (2009). Innovations in insuring the poor: Experience with weather index-based insurance in india and malawi. *Vision for Food, Agriculture, and the Environment*, 17(7):1–2. Available at https://ebrary.ifpri.org/digital/collection/p15738coll2/id/22037.

Gine, X. and Yang, D. (2009). Insurance, credit, and technology adoption: Field experimental evidence from malawi. *Journal of Development Economics*, 89:pp 1–11. Available at https://www.sciencedirect.com/science/article/abs/pii/S0304387808000898.

Gojare, S., Joshi, R., and Gaigaware, D. (2015). Analysis and design of selenium webdriver automation testing framework. *Procedia Computer Science*, 50:341–346. Big Data, Cloud and Computing Challenges. Available at https://www.sciencedirect.com/science/article/pii/S1877050915005396.

Guo, Y. and Liang, C. (2016). Blockchain application and outlook in the banking industry. *Financial Innovation*,, 2:pp 1–12. Available at https://doi.org/10.1186/s40854-016-0034-9.

Hochrainer-Stigler, S., Velde, M., Fritz, S., and Pflug, G. (2014). Remote sensing data for managing climate risks: Index-based insurance and growth related applications for smallholder-farmers in ethiopia. *Climate Risk Management*, (59):pp 27–38. Available at https://doi.org/10.1016/j.crm.2014.09.002.

Iizumi, T. and Ramankutty, N. (2015). How do weather and climate influence cropping area and intensity? *Global Food Security*, 4:46–50. Available at https://doi.org/10.1016/j.gfs.2014.11.003.

Kanjere, J. (2021). A blockchain-enabled system to enhance food traceability in local food supply chains (fscs) suitable for small co-operatives in south africa. *Faculty of Commerce ,School of Economics, University of Cape Town*, pages 1–62. Available at https://open.uct.ac.za/items/9d105e8d-2ecb-4867-b0aa-4991c5c04950.

Kitur, K. K. (2018). Crop insurance strategies for mitigating net underwriting losses. *Walden University*, page 19. Available at https://scholarworks.waldenu.edu/cgi/viewcontent.cgi?article=7016&context=dissertations.

Lichtenberg, E. and Iglesias, E. (2022). Index insurance and basis risk: A reconsideration. *Journal of Development Economics*, 158:1–15. Available at https://doi.org/10.1016/j.jdeveco.2022.102883.

Mapfumo, S. (2007). Weather index insurance the case for south africa. *Micro-Insurance Agency*, pages 1–10. Available at https://www.farm-d.org/app/uploads/2019/05/WeatherIndexInsuranceTheCaseforSouthAfrica.pdf.

Masipa, T. S. (2017). The impact of climate change on food security in south africa: Current realities and challenges ahead. *Jamba (Potchefstroom, South Africa)*, 9(1):1–7. Available at https://doi.org/10.4102/jamba.v9i1.411.

Micali, S. (2016). ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341:1–75. Available at http://arxiv.org/abs/1607.01341.

Miranda, M. J. and Farrin, K. (2012). Index insurance for developing countries. *Applied Economic Perspectives and Policy*, 34(3):391–427. Available at http://www.jstor.org/stable/23273812.

Mkuhlani, S., Crespo, O., Rusere, F., Zhou, L., and Francis, J. (2019). Classification of small-scale farmers for improved rainfall variability management in south africa. *Agroecology and Sustainable Food Systems*, 44:pp 7–29. Available at https://doi.org/10.1080/21683565.2018.1537325.

Mzuku, K. (2019). A decentralised asset registry to expand access to finance for the agricultural sector in south africa. *Faculty of Commerce, African Institute of Financial Markets and Risk Management, University of Cape Town*, pages 1–44. Available at https://open.uct.ac.za/items/08d038ff-1f1a-47cd-a9a0-1e6fd8416c15.

Porter, J. (2023). Principles of user interface design. http://bokardo.com/principles-of-user-interface-design/. [Accessed: March 2, 2023].

Robles, M. et al. (2021). Agricultural insurance for development: Past, present, and future. *Agricultural*

development: New perspectives in a changing world, pages 563–594. Available at https://doi.org/10.2499/9780896293830_17.

Sakthi, U. and DafniRose, J. (2022). Blockchain-enabled smart agricultural knowledge discovery system using edge computing. *Procedia Computer Science*, 202:73–82. International Conference on Identification, Information and Knowledge in the internet of Things, 2021. Available at https://doi.org/10.1016/j.procs.2022.04.011.

Shrivastava, S. and Prapulla, S. (2020). Comprehensive review of load testing tools. *International Research Journal of Engineering and Technology*, 7:1–4. Available at https://www.irjet.net/archives/V7/i5/IRJET-V7I5651.pdf.

Tadesse, M. A., Shiferaw, B. A., and Erenstein, O. (2015). Weather index insurance for managing drought risk in smallholder agriculture: lessons and policy implications for sub-saharan africa. *Agricultural and Food Economics*, 3(1):26. Available at https://doi.org/10.1186/s40100-015-0044-3.

Tsikirayi, C. M. R., Makoni, E., and Matiza, J. (2013). Analysis of the uptake of agricultural insurance services by the agricultural sector in zimbabwe. *Journal of International Business and Cultural Studies*, 7:1–14. Available at https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4febb01b3c8261db770e3ccdcb29f8a8fdb70cb4.

Weber, E. J. (2019). Weather index insurance in sub-saharan africa. *The University of Western Australia - UWA Business School*, pages 1–11. Available at http://dx.doi.org/10.2139/ssrn.3396489.

Yaga, D., Mell, P., Roby, N., and Scarfone, K. (2019). Blockchain technology overview. *CoRR*, abs/1906.11078. Available at http://arxiv.org/abs/1906.11078.

Yoo, S. and Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability*, 22(2):67–120. Available at http://dx.doi.org/10.1002/stvr.430.

APPENDIX

A. Weather Data Collection

This appendix shows the weather data collection components.

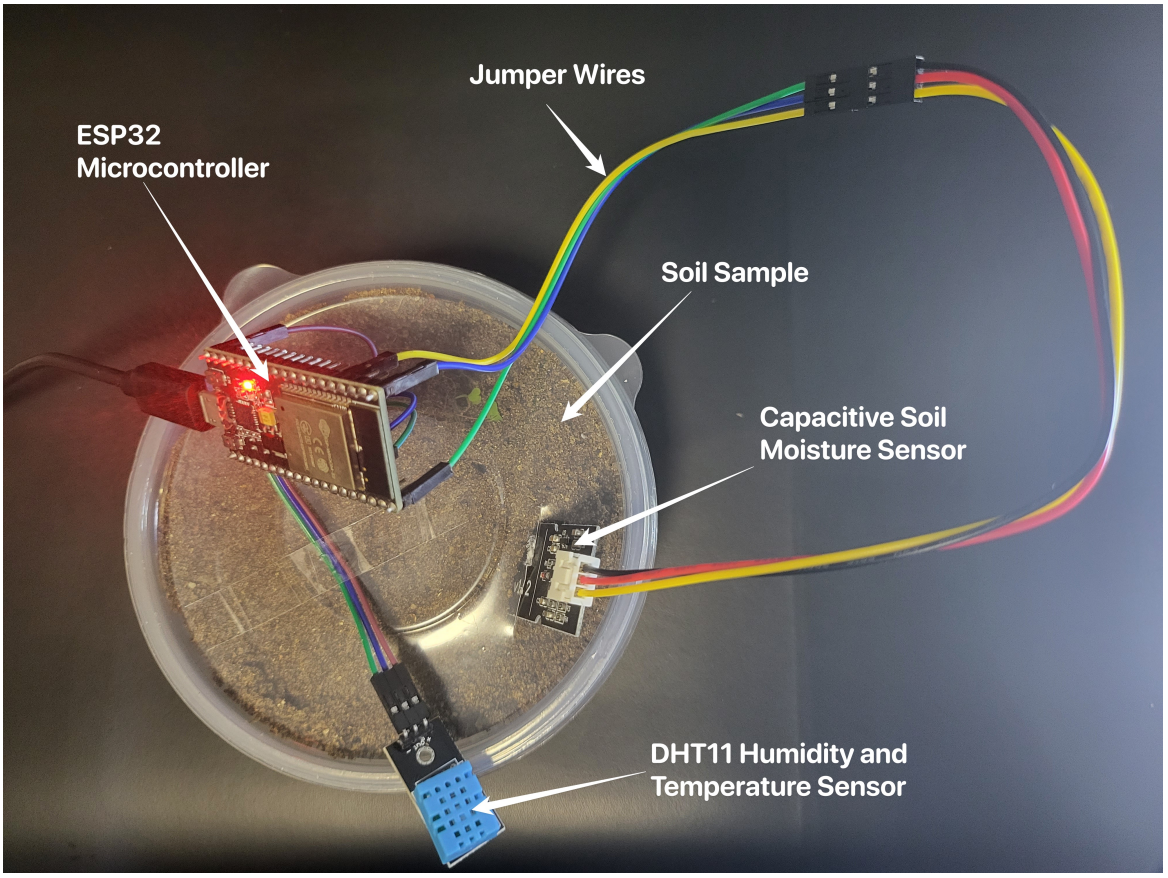

Figure 34: Weather data collection system setup showing circuit connections between the ESP32 microcontroller, DHT11 humidity and temperature sensor, soil moisture sensor and a soil sample used to measure soil moisture. The soil moisture sensor is inserted into the soil and connected to the ESP32 microcontroller via jumper cables. The DHT11 humidity and temperature sensor is also connected to the ESP32 microcontroller via jumper cables. The system is powered by connecting the ESP32 microcontroller to a battery power source. The microcontroller collects humidity and temperature readings from the DHT11 humidity and temperature sensor connected to digital pin 5. Soil moisture readings are collected from the soil moisture sensor connected to analog to digital pin 34. The sensors are powered by connecting their VCC and GND pins to the microcontroller's 3.3V and GND pins respectively.

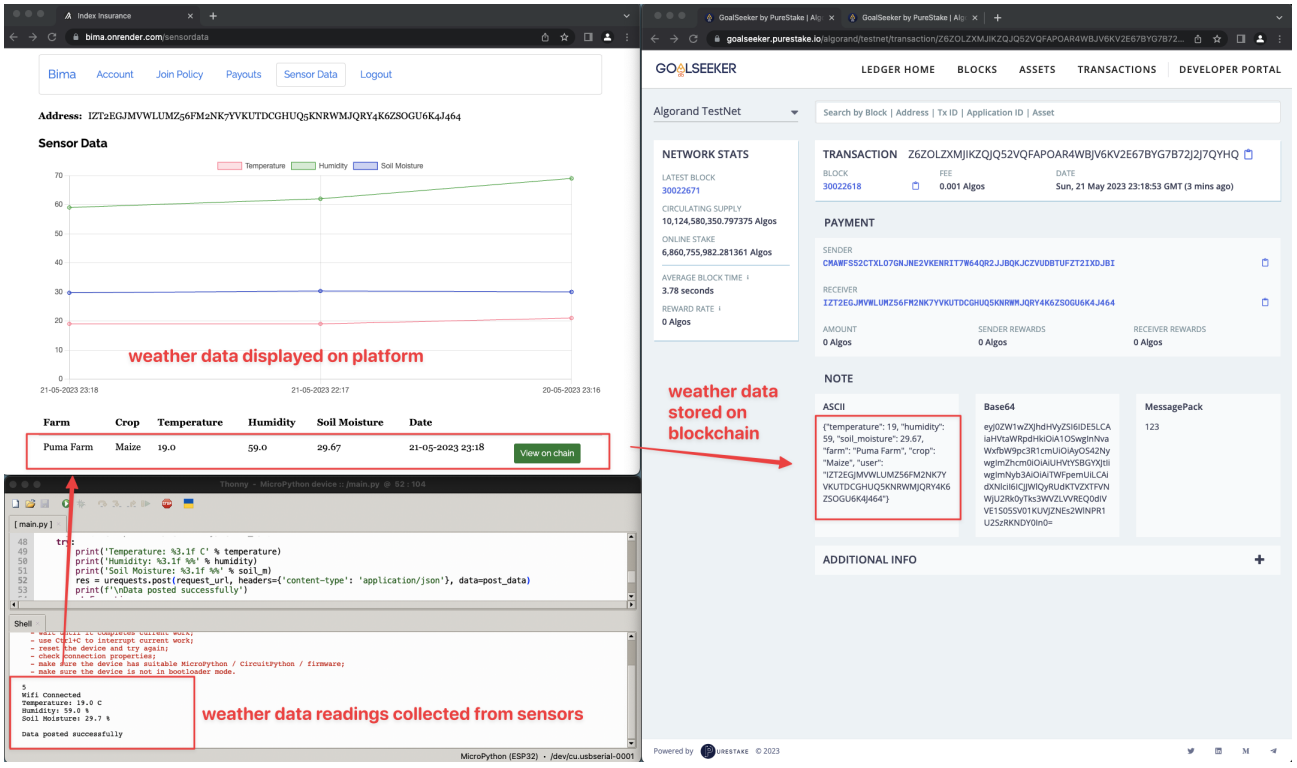

Figure 35: Weather data logging process, showing the microcontroller connecting to Wifi and posting data to the platform's weather data endpoint, the record being displayed on the sensor data page as well as being logged on the blockchain.

B. Load Testing Results

This appendix shows the results of load testing the system using Locust. The endpoints tested are:

- https://bima.onrender.com/joinpolicytest - simulates farmers joining policies.
- https://bima.onrender.com/login - simulates farmers logging in on the platform.
- https://bima.onrender.com/payouttest - simulates weather data being sent from sensors and insurance payouts being triggered when strike conditions attached to insurance policies are met.

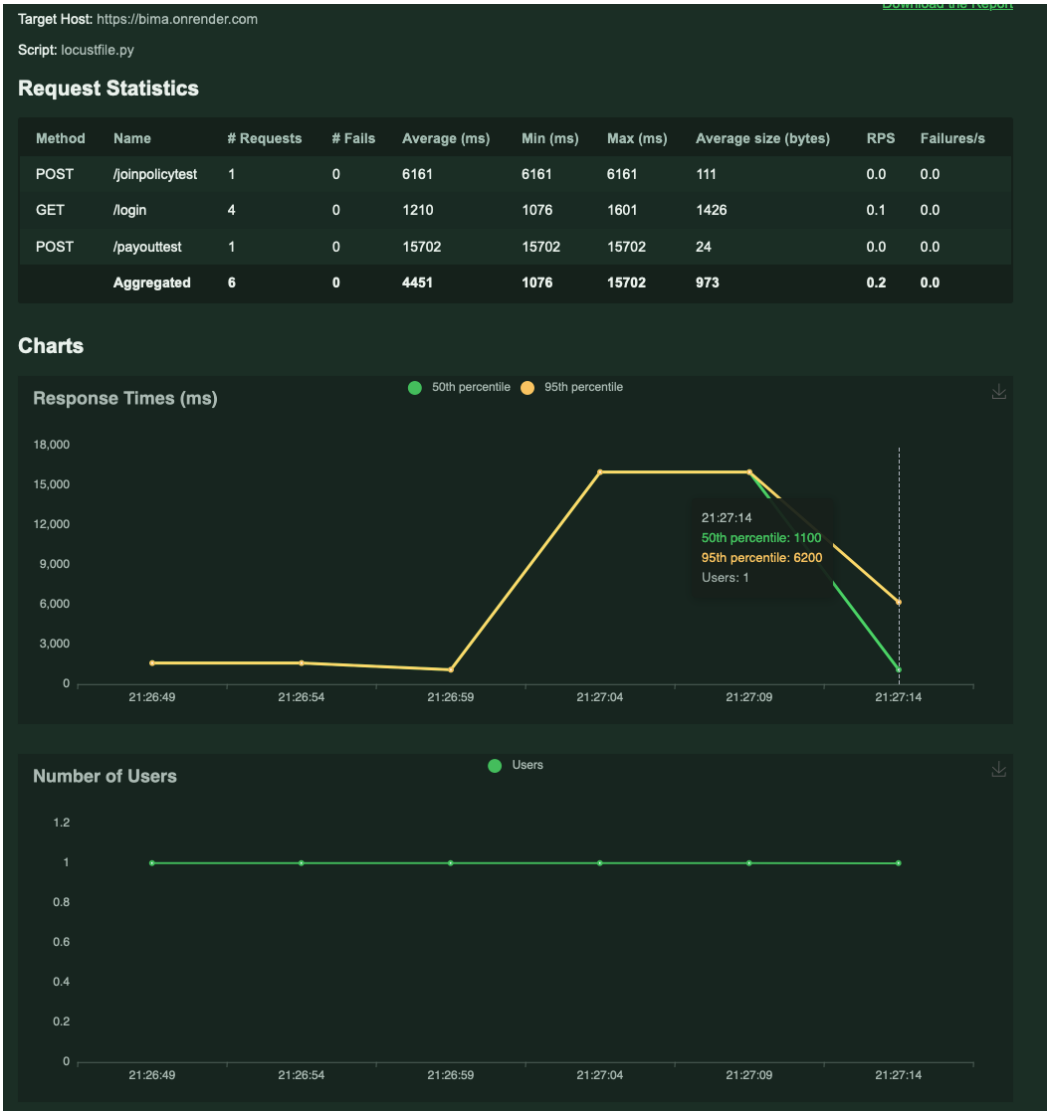

Figure 36: Platform load testing results without load balancing after testing the system with 1 user. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

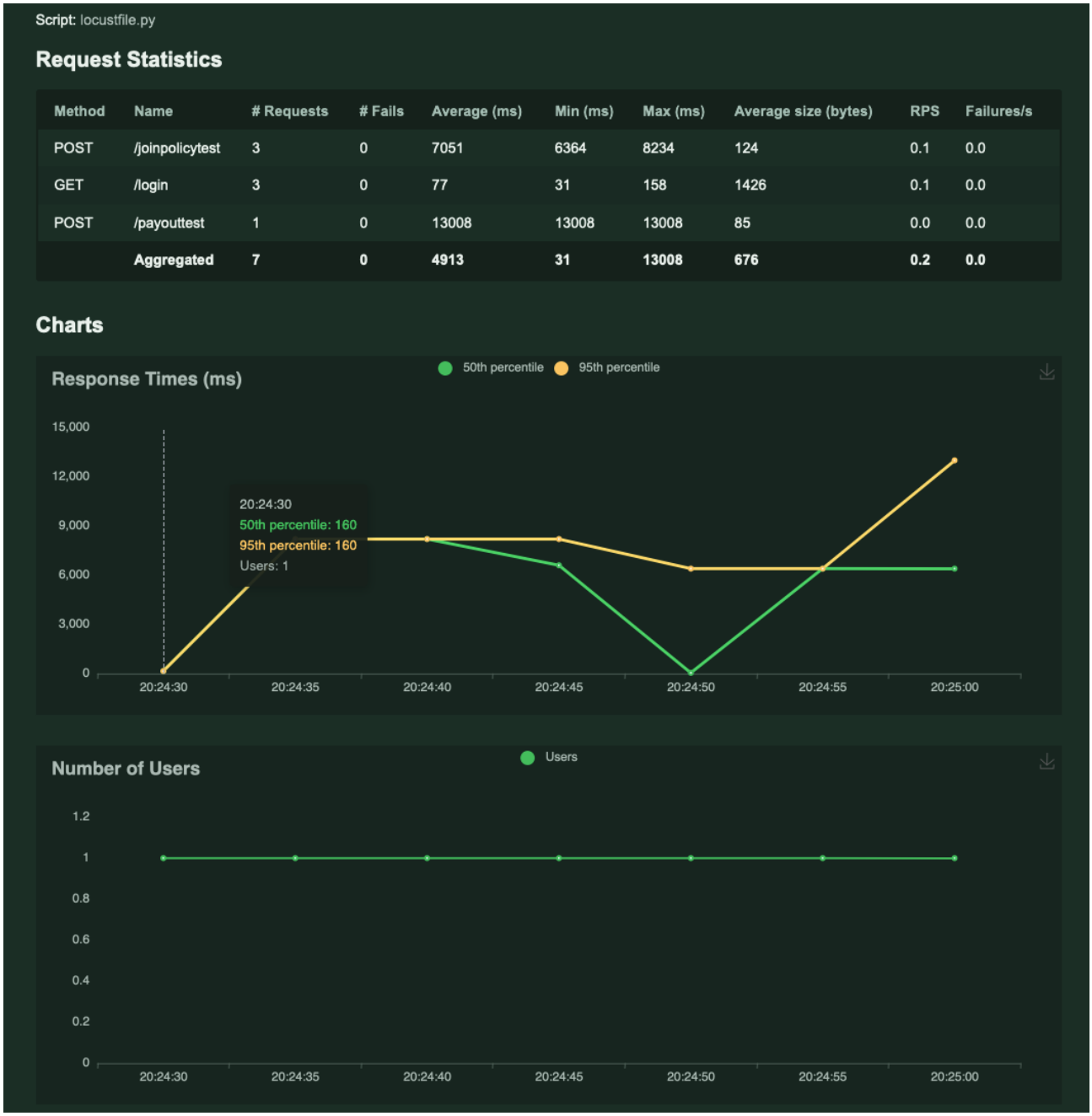

Figure 37: Platform load testing results with load balancing after testing the system with 1 user. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

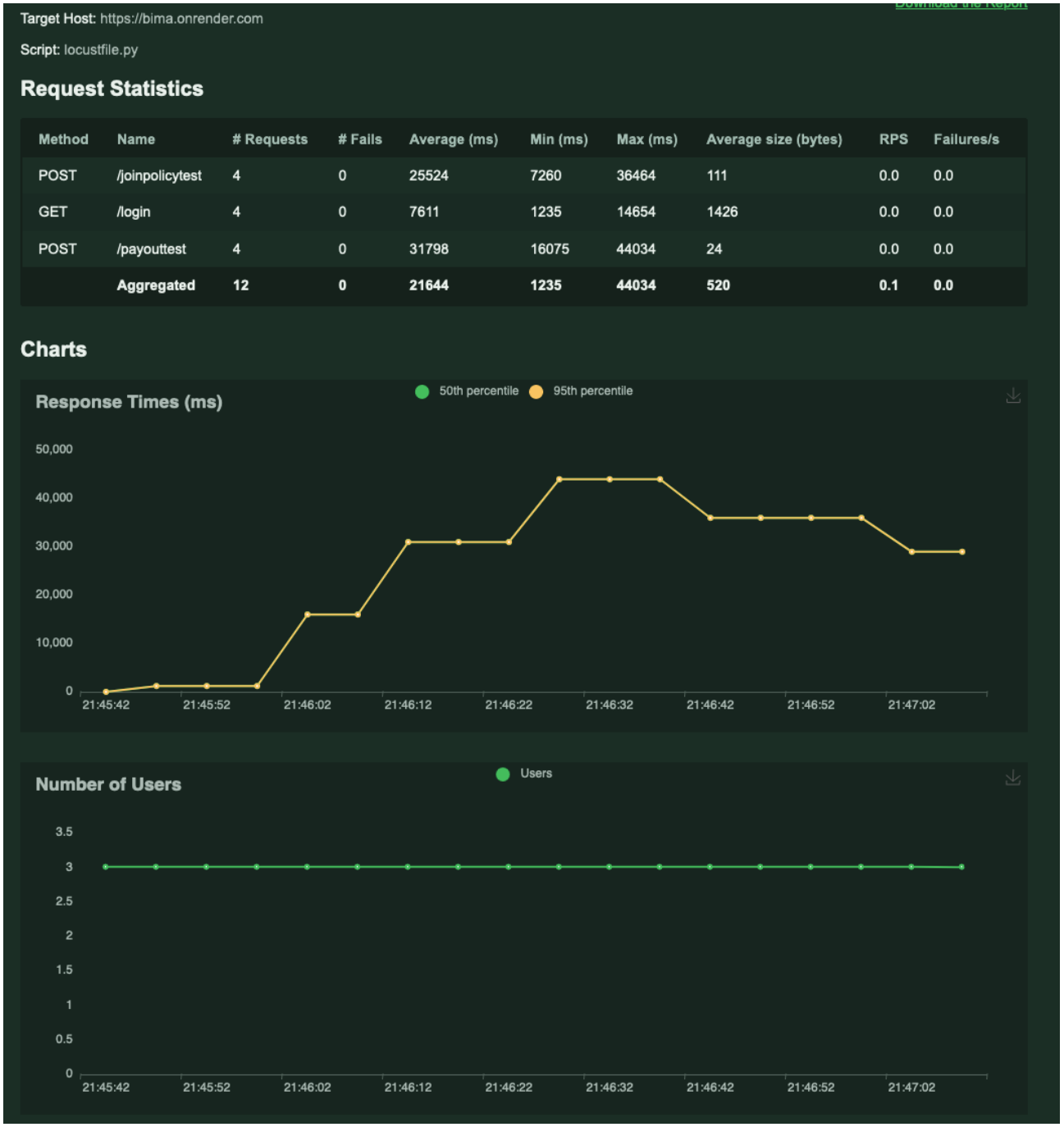

Figure 38: Platform load testing results without load balancing after testing the system with 3 users. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

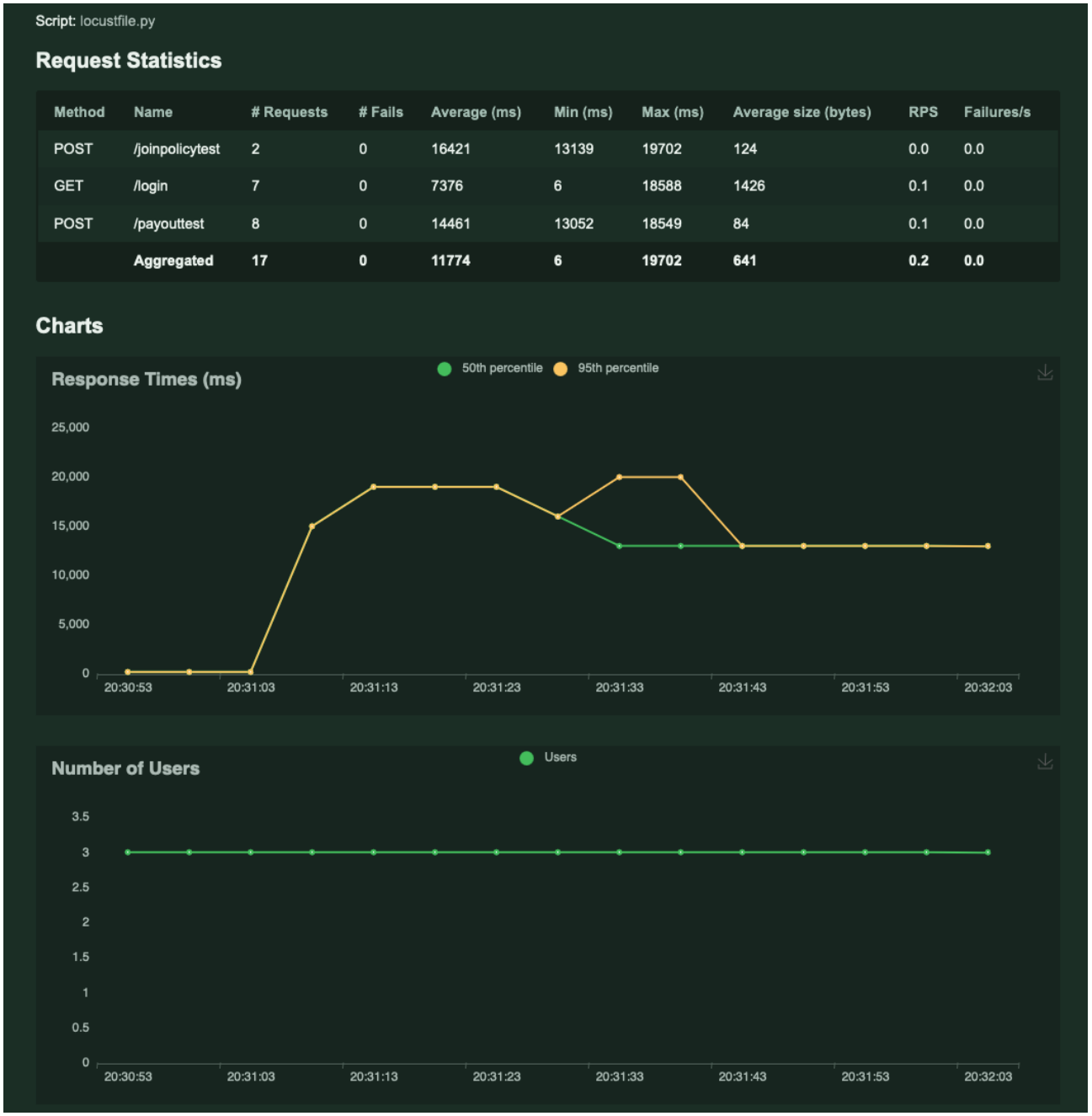

Figure 39: Platform load testing results with load balancing after testing the system with 3 users. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

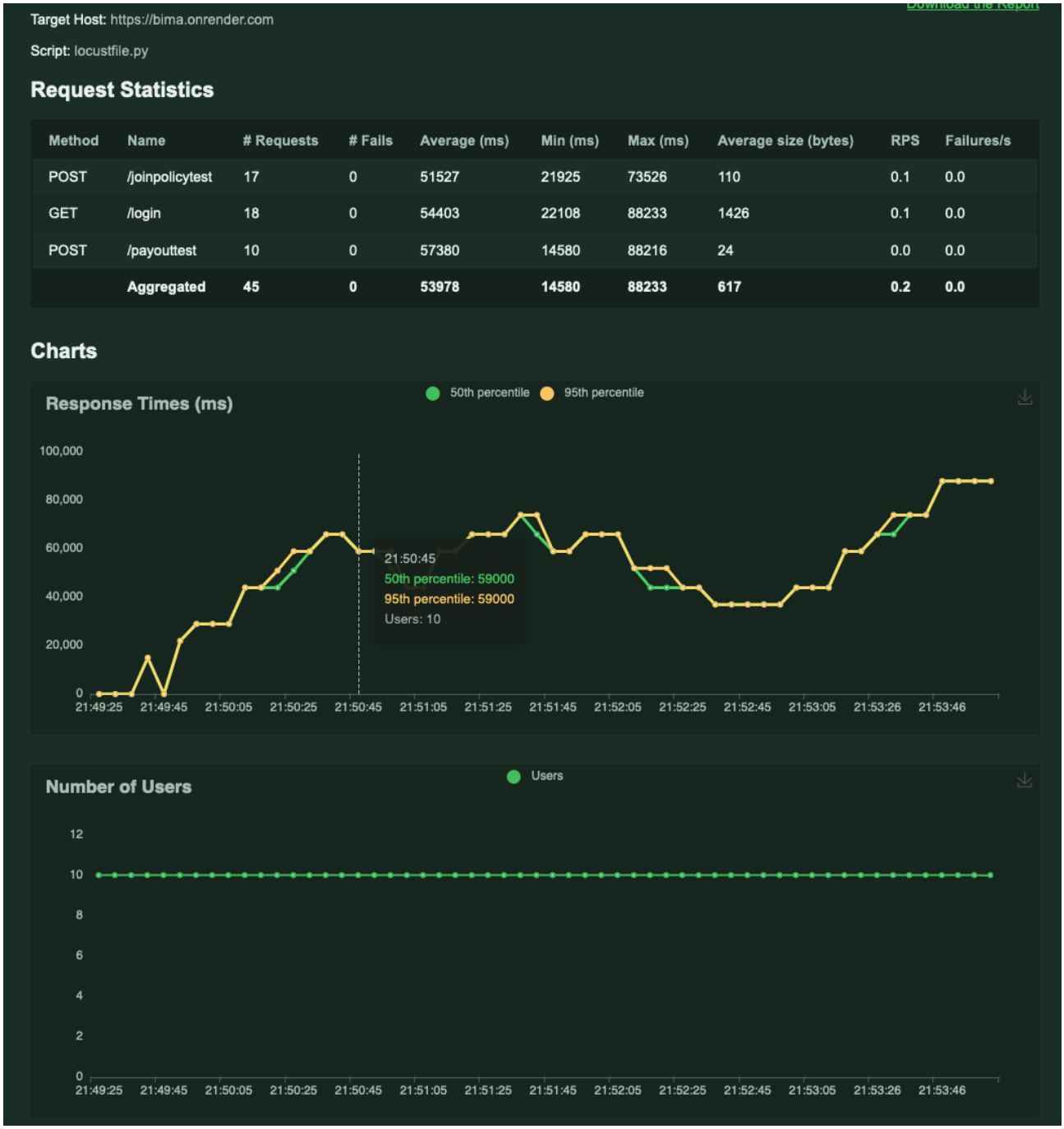

Figure 40: Platform load testing results without load balancing after testing the system with 10 users. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

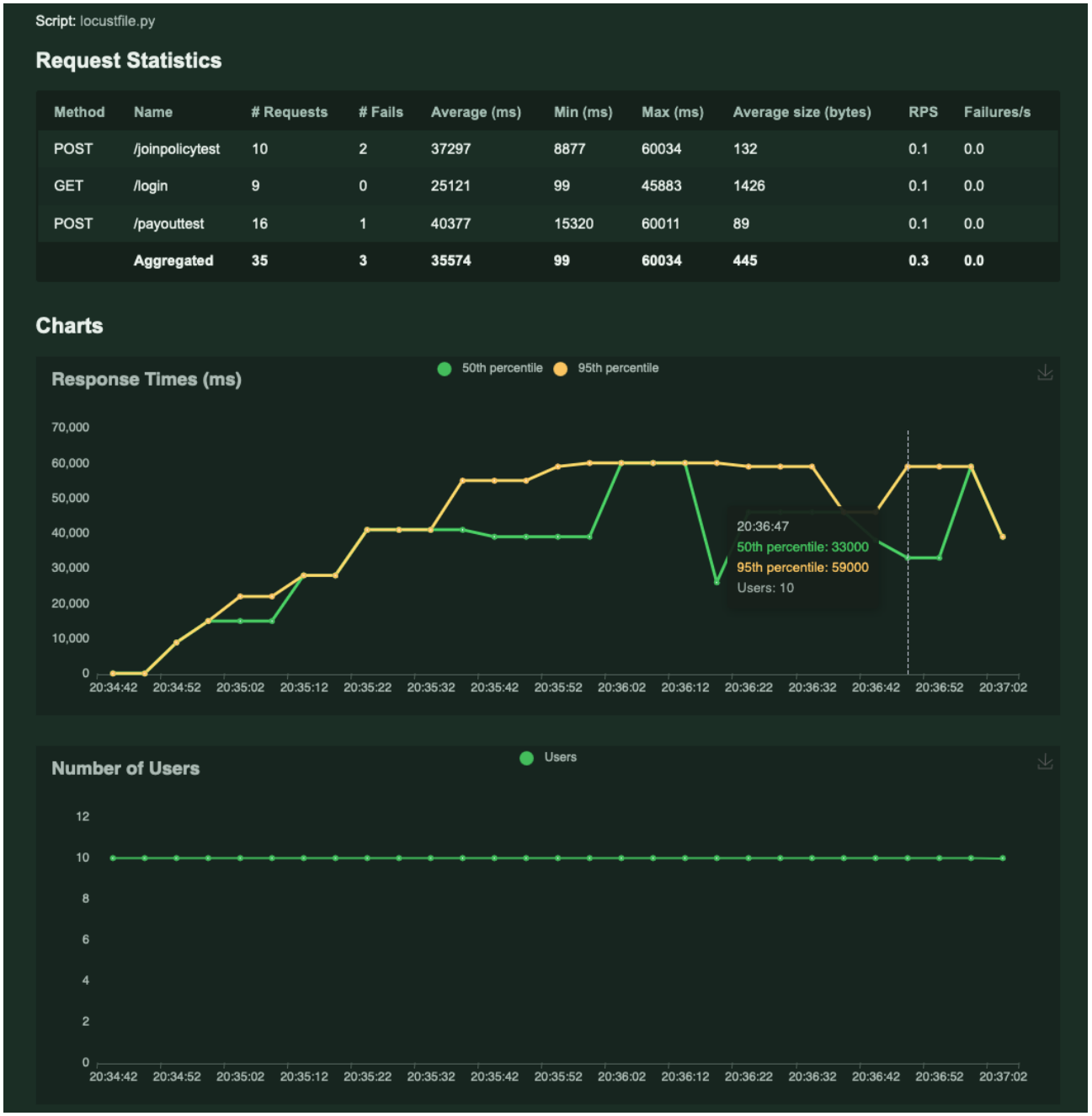

Figure 41: Platform load testing results with load balancing after testing the system with 10 users. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

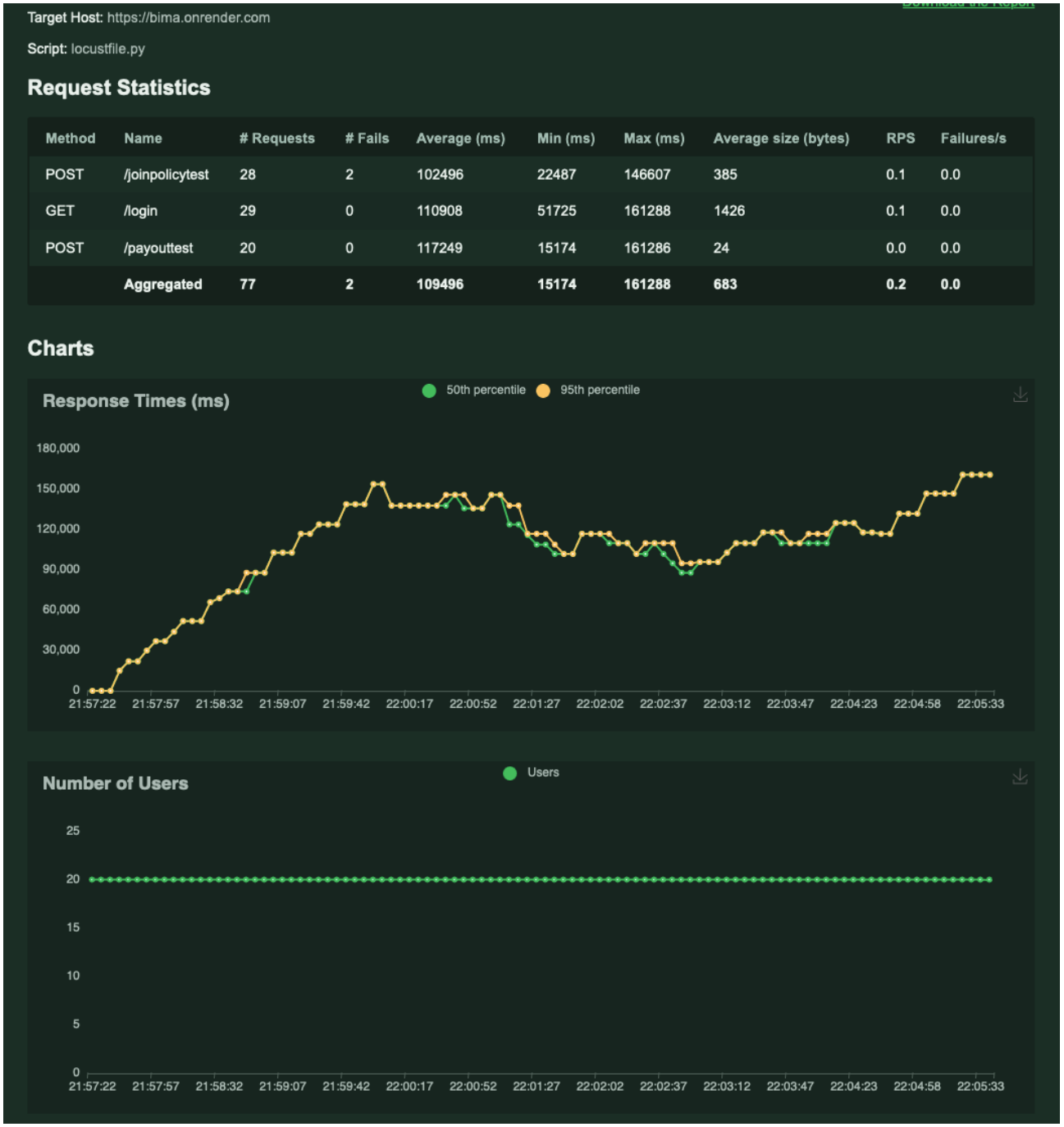

Figure 42: Platform load testing results without load balancing after testing the system with 20 users. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.

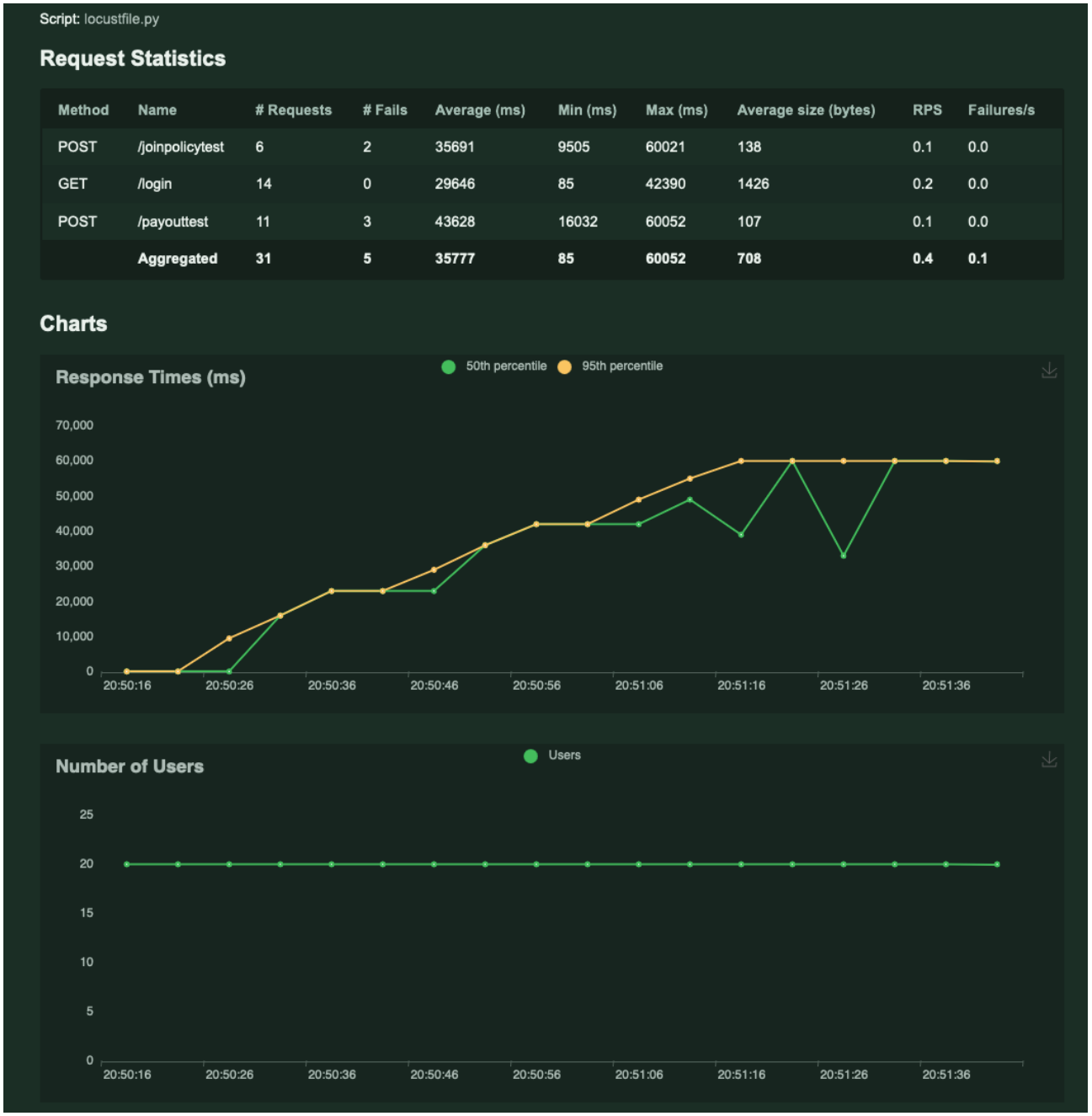

Figure 43: Platform load testing results with load balancing after testing the system with 20 users. The request statistics table shows the average, minimum and maximum response times from the joinpolicytest, login and payouttest endpoints.