Grant Proposal: `Aquila - parametric forest fire insurance using Chainlink oracles`

Name of Project: `Aquila`

Proposal Category:
`prototype`

Proposer: `1eskor`

(Optional) Technical Sponsor:
`-`

LICENSE Do you agree to open source all work you do on behalf of this RFP and dual-license under MIT and APACHE2 licenses?: `Yes`

Project Description

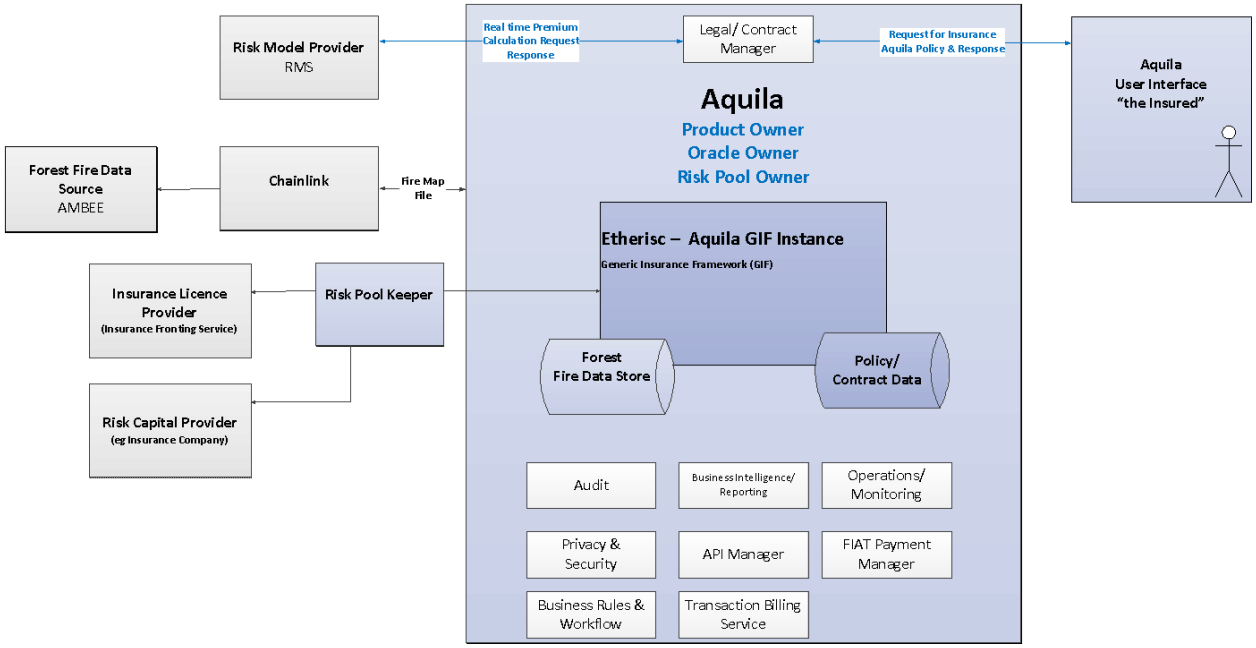
```
Please describe exactly what you are planning to build. Make sure to include the following:
- Start with the need or problem you are trying to solve with this project.
- Describe why your solution is going to adequately solve this problem.

This section should be 2-3 paragraphs long.
```
Aquila is focused on the creation of parametric forest fire insurance through smart contracts. We plan on launching the first parametric forest fire insurance application in British Columbia, Canada. The forest fire insurance industry in Canada is currently valued at a few billion dollars per year, projected to double by 2028 due to climate change. Our solution optimizes the insurance process, by providing a more transparent and fair system for users. It also expedites the claims/settlement process, and provides significant security benefits for payouts. Parametric insurance is a non-traditional insurance product that offers pre-specified payouts based upon a trigger event. In the case of our forest fire application, our trigger event is a forest fire event crossing a pre-set property area. Aquila innovates the claims processing by comparing current satellite images of burning areas against specific coordinates of insured land. This allows for automating and expediting the adjudication and payout process. Chainlink is used to get the external data securely into the smart contracts, which provides decentralized oracle network middleware.

Smart contracts enable the creation of virtual trusted third parties that will behave according to arbitrary agreed-upon rules. There are numerous unique advantages that blockchains contain that traditional agreements do not provide today. The decentralization among nodes in the network creates a tamperproof, trust-minimized, and deterministic environment, which enables greater efficiency, transparency, and mitigated counterparty risk. What this means is no one party, no one implementer, no one validator, no one government is in a position to compromise

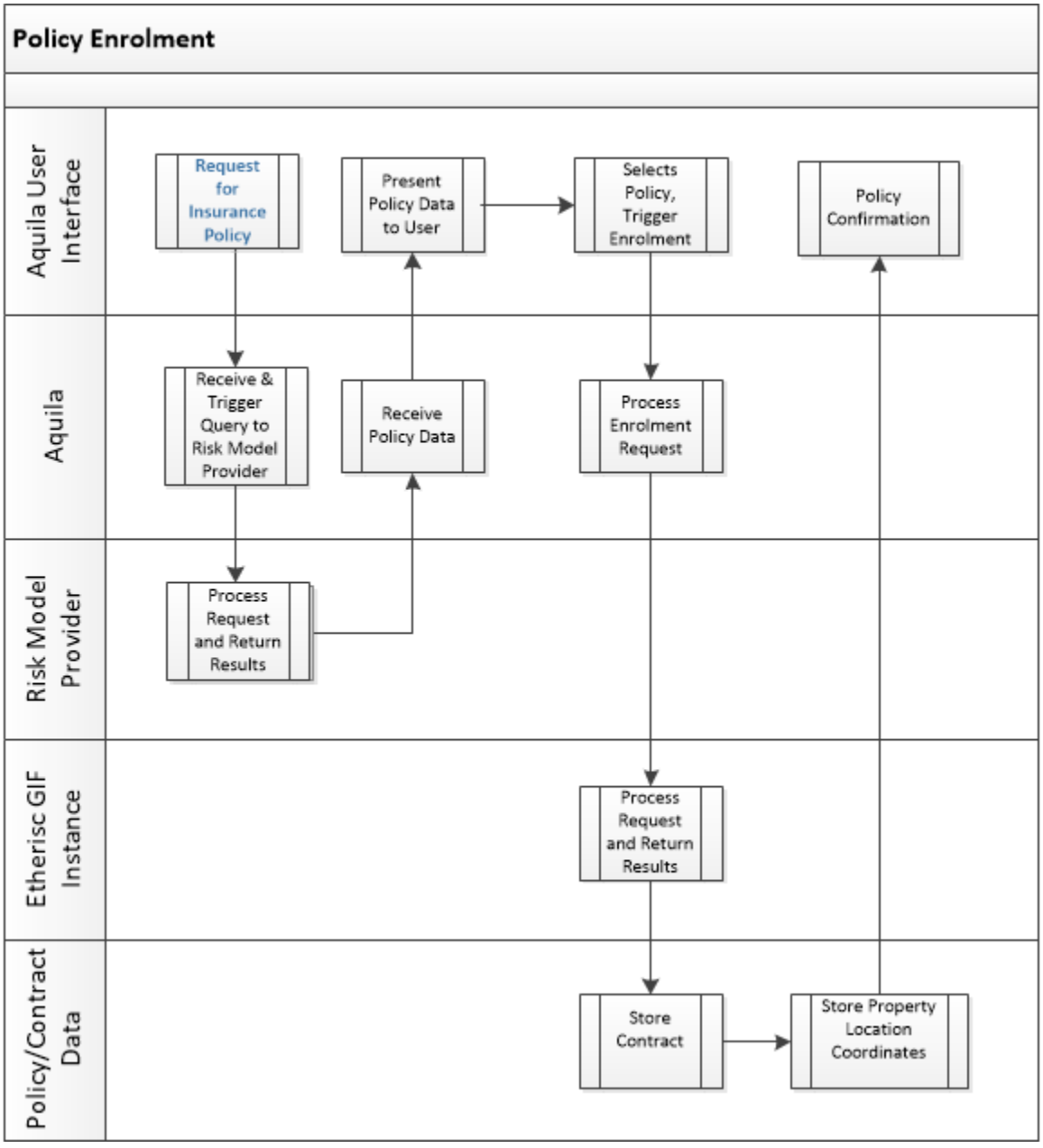
the integrity of that system. If there are multiple validations of something, they're all adding to the correctness and our assurance of the correctness of the data.

In the context of insurance, there is extraordinary value to be enabled by these systems. We believe that enabling people to interact with these novel digital permissionless systems is where the future is headed, away from the existing acquiescent model. There is good reason for this due to the inherent nature and properties that distributed ledgers permit. What are the technical aspects that set decentralized insurance apart? Many aspects around insurance processes can be automated by smart contracts. A policyholder's wait time for processing usually takes a lengthy period of time and is an aggravating process to endure. The automation of claims settlement enables faster/near instant payout. The expenses of traditional claims management are almost entirely removed, thus allowing the issuance of cheaper policies. Users have a greater degree of security of payout through codified trust-minimized contract execution, instead of trust based paper agreements. Additionally Blockchains eliminate conflict of interest between counter-parties, along with providing transparent and objectively verifiable data. Along with those benefits, there is the opportunity to democtritize risk pools to individual investors. We leverage web3 technology in order to create a claims-free process. Aquila pays out automatically based on forest fire damage using aggregated results of the highest quality data providers.

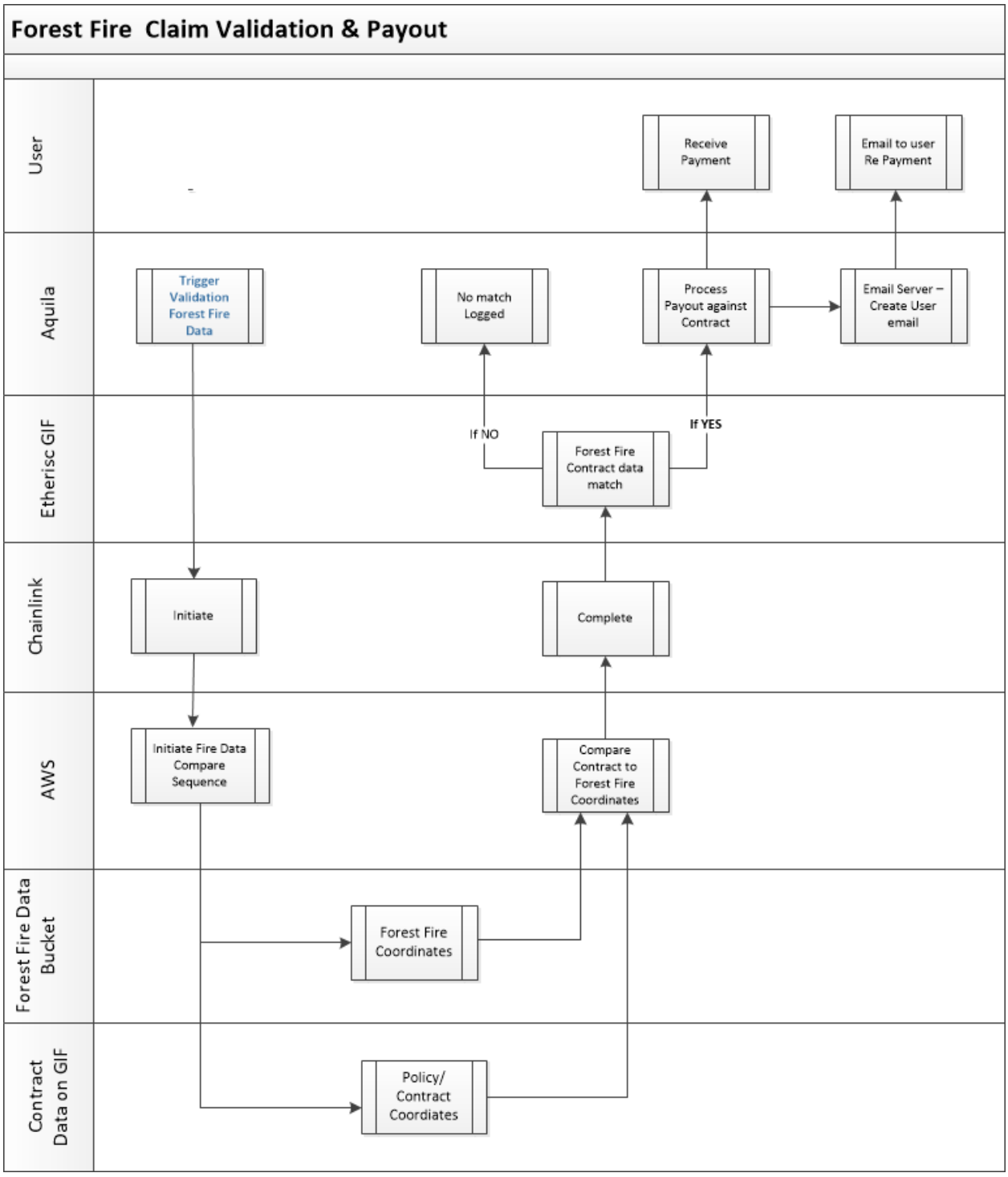
Solution Overview:
This provides a high level overview of the proposed solution, and the partners, integrations and system components that will be implemented for this project. Many components such as operations/monitoring, BI Reporting and billing will be implemented using open source products.



There are two main workfows; enrollment and validation, and payout of insurance claims. Sequence diagrams are included to provide a better understanding of each.

```



```

Claim Validation and Payout



Value

```
Please describe in more detail why this proposal is valuable for the Chainlink and Etherisc ecosystem.
Answer the following questions:
- What are the benefits to getting this right?


The value to the ecosystem is the immense potential of bringing a portion of the multi-billion dollar forest fire market on-chain through Etherisc and Chainlink. As well as the potential growth opportunities as we secure insurance premiums on chain that will utilize the Chainlink oracles and Etherisc platform. This opens the door for other new business opportunities to be presented to clients who are part of the network and this will ultimately expand Etherisc reach. For Chainlink, it adds to the TVL of the network by reliably supplying data to contracts.

- What are the risks if you don't get it right?
This depends on the flaws that exist, theoretically there are many substantial risks, such as not executing the contract correctly, or some exploit or hack that leads to funds being siphoned. Particularly in the last example, we can mitigate the ability for the smart contracts funds to be maliciously utilized, by abstracting the control over the funds to an escrow layer, i.e Agoric. Which would drastically reduce the outcome of damages to a modicum of possibilities. Zoe guarantees that as a user of a smart contract, you will either get what you wanted or get a full refund, even if the smart contract is buggy or malicious. (In fact, the smart contract never has access to your digital assets.) In most smart contracts that are not built with Agoric, the assets put into escrow, are put into escrow with the contract itself. Therefore a buggy or malicious contract can cause those assets to be lost. Zoe is an intermediary between the participants in the contract and the contract itself. And Zoe enforces “offer safety”. Agoric could be migrated at a later date, since it is still in beta. (more info in the additional information section). The initial implementation can be written with solidity operating on a EVM chain, and so long as audits, and safe code has been written, the deployment of the contract can function as intended.  In terms of reducing the risk of the smart contract not executing correctly, this comes down to how well the code has been defined, and if all the necessary protocols for reducing areas of attack have been implemented. An example of how we could reduce the likelihood of a trigger-event malfunctioning, would be correctly adhering to the decentralized data source model utilizing Chainlink as the trust aggregation layer. In addition to multiple data sources, the security of oracle networks (such as Chainlink) depends upon having higher-level contracts to aggregate the results of the individual nodes. This protects against mis-behaviour from an individual node. Relying on just a single node can be both expensive and risky. Instead, use the higher-level APIs described in the Chainlink integration documentation.

- What are the risks that will make executing on this project difficult?
```

liability and responsibility, making sure that there are no vulnerabilities, exploits, or hacks possible within the contract. This can be mitigated by ensuring that adequate measures are in place for upgrading/maintaining the system. Reliability must be a key aspect of the design.. Including a thorough audit to uncover any defects before launch.

This section should be 1-3 paragraphs long.
```

## Deliverables

```
Please describe in details what your final deliverable for this project will be.
Include a specification of the project and what functionality the software will
deliver when it is finished.
```
The final deliverable for Aquila, will be the product launch on Etherisc. The product will include the implementation of all the functionalities needed to complete a full end-to-end parametric insurance contract. There are 3 core aspects of deploying an Etherisc product; technical framework, legal framework, risk transfer market. Part of the technical framework includes oracles i.e Chainlink. Other parts of the product not directly correlated to Etherisc are the risk model provider, data provider(s),investor capital, Aragon. Insurance licenses will be categorized under the legal framework. All of this code will exist on top of an l1 such as Eth, Xdai, ect.

https://docs.google.com/spreadsheets/d/159hoDLEwqndjKZE1LxeXFoDIIRzt3yhx/edit?usp=sharing&ouid=101420523597744295062&rtpof=true&sd=true

**Legal framework:**

Through the utilization of the captive model, it allows us to control the pool of funds. If it's depleted, reinsurance would kick in, there's not a deductible unless its structured that way. Essentially as a captive, we would write our own manuscript policy wordings, the introduction, the insuring agreement and exclusions, general conditions, definitions and claims process which would be the written/paper version of the smart contracts. The captive approach also gives us authority to underwrite policies.
As an advantage of this model, any surplus funds that exceed the expected losses in the 3-5 year range can go 50% towards raising the reinsurance entry point, I.e. the larger the primary pool, the less reinsurance as that comes at a higher premium; and 50% back to policyholders in the form of dividends. In a DeFi protocol like Etherisc it could be in the form of DIP tokens. There’s flexibility to also divide the surplus in many ways: primarily pool, investments, policyholder and service providers, especially to the distribution channels to boost sales.
```

Development Roadmap

```
Please break up your development work into a clear set of milestones.
This section needs to be very detailed (will vary on the project,
but aim for around 2 pages for this section).

For each milestone, please describe:
- The software functionality that we can expect after the completion of each milestone.
  This should be detailed enough that it can be used to ensure that the software meets
  the specification you outlined in the Deliverables.
- How many people will be working on each milestone and their roles
- The amount of funding required for each milestone
- How much time this milestone will take to achieve (using real dates)

https://docs.google.com/spreadsheets/d/159hoDLEwqndjKZE1LxeXFoDIIRzt3yhx/edit?usp=sharing&ouid=101420523597744295062&rtpof=true&sd=true
```

Development Roadmap - Aquila, Chainlink/Etherisc Grant

Deploy Etherisc GIF Instance - Establish Core Infrastructure. Work with Etherisc published specifications; seek support where required to install a GIF instance in the dev and test environement.

Risk Transfer Market:
As described in the Etherisc users/contributors pdf, risk capital providers can be individuals, hedge funds, or institutions. In our proposal, we iterate that our risk capital provider will be an institution AKA reinsurance company. This is in line with a typical insurance model, a % of the transaction fee is sent to the reinsurance premium, which could be the same entity providing the insurance license to issue policies, or a separate entity which is legally entitled to cover reinsurance claims. They provide the capital to payout claims in the event of the primary risk pool is depleted. It is worth noting that a certain percentage of DIP tokens must be staked by the risk capital providers.

Legal Framework:
Regulation ensures that the policyholder receives the proclaimed coverage amount in the event of an insurance claim. In one approach, on the insurance side, in exchange for providing the insurance license (the legal obligation to cover the risk, for regulatory compliance) the insurance company receives 3-7% of the premiums collected. For distribution there would be collaboration with a MGA, in order to leverage their large userbase that already exists (which would be the same in both approaches). Since we are operating in Canadian jurisdiction, the safest and most straightforward approach while offering insurance to business's or individuals is to have the licenses required, despite 'blockchains not operating within any particular region', the laws still exist to protect policy buyers.
We are taking into consideration various approaches, either through the standard insurance company, a protected cell captive or segregated cell captive. The 3 approaches have their own distinct advantages, the main advantage with the captive model is the more 'decentralized' approach that allows a distribution of dividends to members from excess funds, instead of the insurance company taking all the profit. Through the utilization of the captive model, it allows us to control the pool of funds. If it's depleted, reinsurance would kick in, there's not a deductible unless its structured that way. As a captive, we would

write our own manuscript policy wordings, the introduction, the insuring agreement and exclusions, general conditions, definitions and claims process which would be the written/paper version of the smart contracts. The captive approach also gives us authority to underwrite policies.
As an advantage of this model, any surplus funds that exceed the expected losses in the 3-5 year range can go 50% towards raising the reinsurance entry point, I.e. the larger the primary pool, the less reinsurance as that comes at a higher premium; and 50% back to policyholders in the form of dividends. In a DeFi protocol like Etherisc it could be in the form of DIP tokens. There's flexibility to also divide the surplus in many ways: primarily pool, investments, policyholder and service providers, especially to the distribution channels to boost sales.

Technical Framework/Generic Insurance Framework (GIF):
The GIF consists of 3 components, which come together to form a GIF instance, which must be approved by the instance operator. The main elements are product, oracle, and risk pools, Aquila plans to own/manage the 3 different components.
Risk pool managing is done by 'risk pool keepers', who control the smart contract of the risk pool consisting of policy objects and risk capital.
Oracles are used to provide external data to smart contracts, in our case, current fire data. Chainlink's flexible framework allows us to customize which data sources and nodes will be returning the data. Nodes reference data feeds about insurable events and therefor allows the creation of an automated claims process. There will be changes made available to the DON of node operators around the data delivery once staking is released.

+ AWS EC2 instance/GCP for Chainlink node

+ AMBEE integration into lambda + s3

The 'product' is the smart contract that encapsulates the capabilities of the dapp, built with either the default options in the GIF, or custom abilities. In our example, since we plan to integrate with Aragon, and Gnosis, custom functionality will be included.

Integration with Risk Model Provider:
Core Infrastructure - establish an interface to a risk model provider. This allows Aquila to query and obtain insurance quotes that can then be displayed in the UI and used in the creation of the insurance policy/contract. RMS catastophe modelling provides real time analysis of risk profiles.

Development & Deployment of Contract Module:
Consumes data from the Risk Model Provider and UI to create the contract resource that is then stored. May include consent; compliance to privacy law for private data. This includes requirements, analysis, design, data model, development, testing and deployment. Connecting the users information to the risk model provider, receiving the quote, sending back to the user to accept.. if accepted, the confidential data stored privately. Aquila will have to accept or deny each application before underwriting the policy, which would confirm that the information of the users identity, ownership of the property, ect is correct, as part of the policy lifecycle.

User Interface:
User interface will present data from the risk model provider (real time interface); allow users to select the policy; trigger creation of a policy, trigger payment collection details, supply obligatory documents, consent/privacy agreement as part of the contract.. User feature to enable them to check their profile with the status of their current contracts.

Development of Payment Module to support payment workflow:
Payment details can be stored as part of the contract, Insurance payouts flow through Aquila form the Risk Pool to client. FIAT gateway for accepting/delivering payments.

Monitoring Software:
Select and deploy open source monitoring softwaree. admin profile, liquidity in the pool, contracts deployed, time expiration of contracts, ect. This is for administrative purposes, details and statistics such as number of policies written, declined policies, expired policies, size of risk pool, payouts to investors, claims under review, correlation of risk model predictions vs paid claims, ect are supplied through this UI.

Managing Liquidity Security:
Risk pool funds must be secured safely. For this functionality we plan to utilize Gnosis safe. The Safe is a smart contract wallet with multi-signature functionality. Multi-signature wallets are contract accounts that require multiple parties to confirm a transaction before it can be executed. These parties, each represented by a unique Ethereum account address, are defined as multi-signature wallet owners in the smart contract. Only when a predefined number of these owners confirm a transaction, will the transaction be executed. Hence, the single point of failure associated with private key-controlled accounts is removed.

Aragon:
governance layer
registry - payouts trigger in batches.. for hurricanes, floods, ect… where 200k policies trigger
you don't want a single account to execute that payout method
solution - multi signature governance via Aragon (for free)
manages rule and governance related to policies and triggers, ex, endpoint, reward %, or other config
there would be an audit trail for those changes

policies - ability to view and generate reports on policies
policies are stores in smart contracts
administrator can see reports and get detailed private information , full list of all policies

disputes management -
single biggest point of failure in automated triggers is the lack of dispute
not just about sensing data,
not having a dispute mechanism reduces the trust in such insurance
solution: multi-sig goverance scheme, where regulators, governments, and relevant stakeholders can scrutinize the decision thats made and decide whether or not there should be an update
these are high stakes contracts, so its imperative that there is a sufficient fallback mechanism in place to overide any potential failures

Total Budget Requested

```
Sum up the total requested budget across all milestones, and include that figure here.
Also, please include a budget breakdown to specify how you are planning to spend these funds.
```
view milestone spreadsheet. *incomplete <~ please comment any suggestions for calculating time and cost*

Maintenance and Upgrade Plans

```
Specify your team's long-term plans to maintain this software and upgrade it over time.`
```

Team

Team Members

```
- Team Member 1 *will add*
- Team Member 2
- Team Member
- ...
```

Team Member LinkedIn Profiles

```
- Team Member 1 LinkedIn profile *will add*
- Team Member 2 LinkedIn profile
- Team Member 3 LinkedIn profile
- ...
```

Team Website

`Please link to your team's website here (make sure it's 'https')` *will add*

Relevant Experience

```
Please describe (in words) your team's relevant experience, and why you think
you are the right team to build this project. You can cite your team's prior
experience in similar domains, doing similar dev work, individual team members' backgrounds,
etc.
```
I got into crypto in 2017, and was fortunate enough to stumble upon Chainlink near the inception. It was very exciting times, as the passionate community was ecstatic about what Chainlink was building, and the future that lied ahead. I was continually day after day researching and discussing Chainlink, and the implications that it would have on society. In 2018, I attended the 2nd ever program for blockchain full stack development at George Brown College. During that semester, I connected with some really great people, and enhanced my back-end knowledge and abilities ten-fold. After school, I continued along my path of persistently learning all I could about blockchain and Chainlink specifically. Eventually, in August 2021, I stumbled across a interview with Sergey and arbol discussing parametric insurance, and heard Sergey mention the use-case of forest fire insurance. About a week later, I witnessed forest fire smoke englufing my new hometown Squamish. I could clearly envision how the application would function, through taking in the Satellite fire data fed to the contract by decentralized oracles, and automatically distributing payments according to whether a property had been destroyed by the wildfire.
I then proceeded to hop in the Chainlink discord, to find someone, who would later become a good friend and team member, to help with the external adapter for bringing the data on-chain. This was the start of the project, deemed Aquila.

The team I am fortunate to work with is also highly qualified. The first team member Jesper (Denmark) I met on Chainlink discord, who immediately recognized the vision for what I was trying to build, and instantly demonstrated his deep understanding of Chainlink and IT infrastructure. Jesper has over a decade of experience working in technology.
Hicham (Germany) is a extremely bright engineer, who is also passionate about web3. Hicham is very skilled in solidity development and we met during the Chainlink fall hackathon.
Chuck (Paris), is working as a blockchain developer, building a lending market for realestate token. Also met during hackathon, and has deep knowledge of Solidity/Web3.
Cielito Villanueva: Insurance advisor, with 15 years experience as a broker, and connections within the insurance industry.
Front End position is currently open.
Anne Belford (Canada) is a solution architect at Telus with decades working within IT.

Other developers include Marcus Wentz (US) experienced blockchain developer with many innovative solutions. And team from protofire.io/ with Renat Khasanshyn the Co-Founder and team member of Etherisc. Protofire is a team of engineers, which helps decentralized protocols and developer platforms to accelerate growth of their ecosystems.

Team code repositories

```
Please provide links to your team's prior code repos for similar or related projects.`
```
https://devpost.com/software/aquila-24osbt

Additional Information

```
Please include any additional information that you think would be useful in helping us
to evaluate your proposal.
```

Users;

Prospect security upgrade:

Agoric;

agoric is a fallback/escrow mechanism if they get a payout or a refund.
even if contract errors,

essentially adding to the safety
and security to the network

it gives guarentees to the contracts;
contracts don’t get the money directly; they don’t have to worry about dropping it or returning it or accidently having someone steal it, so suddenly they owe people money that they don’t have.

Offer Safety ensures:

- to clients:
  - clients recieve desired payout or refund
  -… regardless of the behaviour of the contract

To Contract:

- Offered assets are synchronously available
  - offered assets match their descriptions
  - assets will be delivered upon competition

- it can do a realocation synchronously and it just works

  therefore; it simplifies the architecture of the code;

governance layer
registry - payouts trigger in batches.. for hurricanes, floods, ect… where 200k policies trigger
you don't want a single account to execute that payout method
solution - multi signature governance via Aragon (for free)
manages rule and governance related to policies and triggers, ex, endpoint, reward %, or other config there would be an audit trail for those changes

policies - ability to view and generate reports on policies
policies are stores in smart contracts
administrator can see reports and get detailed private information , full list of all policies

disputes management -
single biggest point of failure in automated triggers is the lack of dispute
not just about sensing data,
not having a dispute mechanism reduces the trust in such insurance
solution: multi-sig goverance scheme, where regulators, governments, and relevant stakeholders can scrutinize the decision thats made and decide whether or not there should be an update
these are high stakes contracts, so its imperative that there is a sufficient fallback mechanism in place to overide any potential failures