Aigang Documentation

Release 2018-11-06

Aigang.network

Dec 19, 2018

Contents

CHAPTER 1

Content

1.1 What is Aigang?

Our goal is to create fully automated insurance for IoT devices and a platform for insurance innovation built around data, predictions, investors and community

To serve insurance to the end user we split our platform into 4 main parts:

- *Data*
- *Predictions Market*
- *Pools*
- *Insurance*

1.1.1 In short

Platform goal is to create balanced Insurance products using this steps:

- **Data** - data provider uploads his data for community for analysis and receive some insurance risk models.
- **Predictions market** - gives opportunity to assess insurance risk

After these steps insurance product will be created. For this kind and for first time insurance product should be created manualy. In creation proces should be created product oracle for claims validations. After product creation these steps continues:

- **Pool** - allows to paricipate into insurance product reserves pool.
- **Insurance** - in this step insurance product is ready for users to start insure they devices.

Insurance product will have life cycles which at the end all left reserves will be distributed for reserve pool participants.

1.1.2 Links

Homepage
Whitepaper
All Aigang.network sources at github.
Platform Web part sources is available here.

1.2 Data

Data part is dedicated for data uploading, analysis and risk models preparation.

1.2.1 Data set

In this space data supplier can upload data and describe data structure for easier understanding. Data set should not contain any sensitive or personal data. File format should fit CSV standard "TODO: here".

1.2.1.1 CSV requirements

- less than 10 Mb
- comma separated
- utf-8 encoding

1.2.1.2 Remote file access point

This field is for describing detailed steps how platform users will be able to access data file and download it. Usually it is set when provided file is too large or contains other extension than .csv

1.2.1.3 Data set state

- Waiting for approval - data set is currently under review by administrators
- Public - data set is visible for all platform visitors
- For logged in users - data set is visible for logged in users only
- Closed - data set is no longer relevant. No edit or new models submission allowed

1.2.1.4 Data set example

```
"DATE","REGION","DEVICEBRAND","DEVICEYEAR","BATTERYWEARLEVEL","ITEMID"
"2017-09-21 16:29:50","unknown","asus","2016","100","bf2405968e460a53"
"2017-09-22 9:35:15","unknown","meizu","2017","90","1002215a3478a570"
"2017-09-22 20:36:15","unknown","pantech","2014","90","93e2b97de177ed99"
"2017-09-23 18:12:20","unknown","lge","2014","90","2d3b8a37258d60e6"
"2017-09-24 7:33:00","unknown","samsung","2016","90","6f45fb11f3c702d9"
```

(continues on next page)

(continued from previous page)

```
"2017-09-24 8:57:26","unknown","huawei","2017","0","1008b1d08d372006"
"2017-09-24 13:17:10","unknown","explay","2014","90","4332da77a0e1f4d2"
"2017-09-24 13:39:48","unknown","samsung","2017","90","3e2d1c3656ab5cc0"
"2017-09-25 1:03:56","unknown","samsung","2014","90","928abd57acf557fa"
```

Each data set can be commented by logged in users.

1.2.2 Data models

Data model is insurance risk model which was made from data set. With data model user align risk for insurance product. "Base premium" value is value from where risk multipliers begins. For more precise models user can create up to 10 risk tables.

Each Data model can be up-voted.

Model with highest vote will be implemented into insurance product and owner will get reward.

1.2.2.1 Data model example

Android Phone battery Insurance:

Base premium: 2 ETH

DEVICEBRAND	Multiplier
asus	1
meizu	1
samsung	1
OTHERS	1.5

DEVICEYEAR	Multiplier
< 2016	1.5
2016	1.3
2017	1.1
> 2017	1.3

Example:

If user has 2017 Samsung mobile device and wants to insure his battery, then "Risk premium" for him will be 2 * 1 * 1.1 = 2.2 ETH

Insurance claim is valid and payout will happen if BATTERYWEARLEVEL is lower than 90.

1.3 Predictions Market

Space for forecasts where users can profit from theyr knowledge or math skils.

1.3.1 Domain language:

- **Prediction** - is a statement about a future event. In the prediction market are stored a lot of predictions with user forecasts.

- **Outcomes**—a list of outcomes available in prediction for users to choose.
- **Forecast**—is a user selected outcome in prediction with tokens amount.
- **Oracle**—contract or API which will know prediction result at the end time.

1.3.2 Successful forecast flow

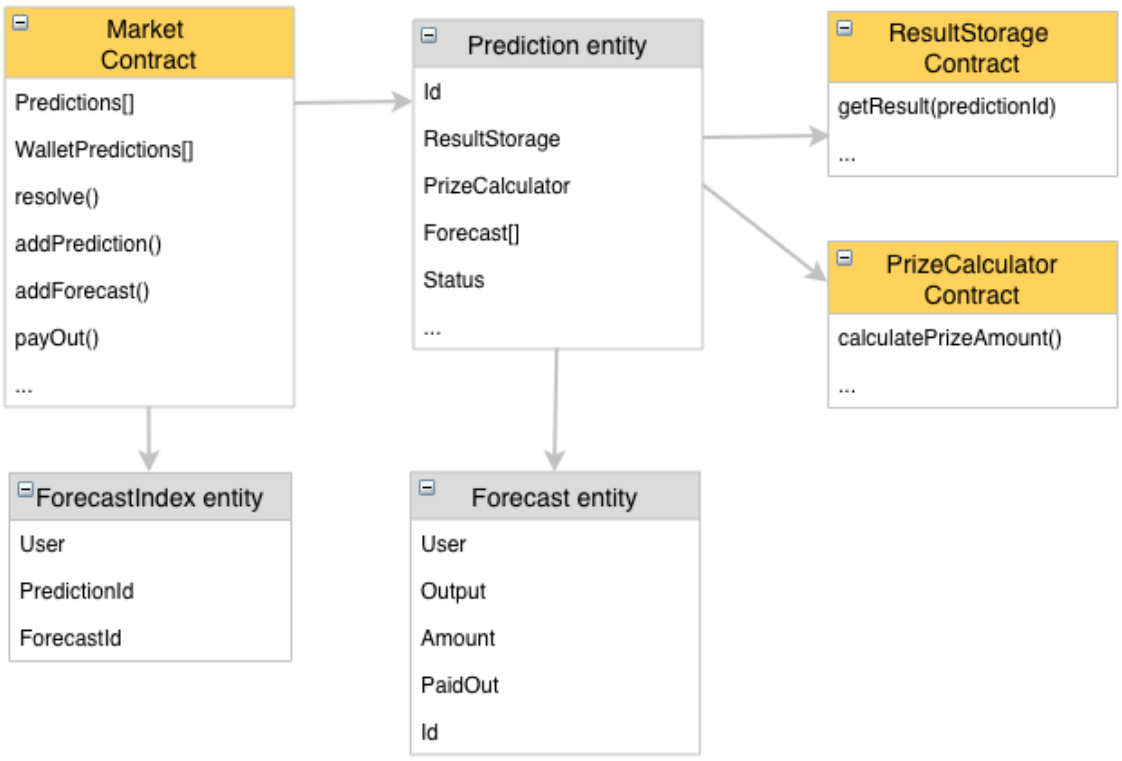
- A user selects active prediction.
- Select his favorite outcome.
- Create a forecast with AIX tokens amount.
- After prediction will be resolved an user will be able to withdraw his winning amount.

1.3.3 More details

1.3.3.1 Prediction Market structure

Blockchain Part

Full view:



Market Contract

The main contract will be **Market** contract which holds:

- AIX Balance of all predictions
- All actions Events
- Available predictions list
- Participants forecasts details

Prediction

The entity holds all participants forecasts and addresses to ResultStorage, PrizeCalculator. Full Entity structure:

- id - 32 symbols hash
- forecastStartUtc - Start date when Prediction starts to accept forecasts.
- forecastEndUtc - End date until which prediction accept forecasts
- fee - Each prediction can have a fee.
- status -
  - NotSet (0)
  - Published (1) - prediction is ready for market. Initial status.
  - Resolved (2) - prediction winning outcome is known and payouts are ready.
  - Paused (3) - participation in this predictions is paused and administrators are investigating what is happening
  - Canceled (4) - some issue happened and refunds for this prediction participants will happen.
- outcomesCount - number how much outcomes are available
- resultOutcome - outcome index which won prediction. Start from 1.
- forecasts - an array of participants forecasts:
  - id - 32 symbols hash
  - user - forecast owner wallet address
  - amount - forecast size in AIX tokens
  - outcomeId - forecast selected outcome index
  - paidout - if the prediction is canceled or forecast won - an amount which was paid for user
- outcomeTokens - array with information how much tokens has each outcome index
- initialTokens - a number of tokens which was transferred by an organizer
- totalTokens - prediction tokens amount
- totalForecasts - forecasts participated in this prediction count
- totalTokensPaidout - total tokens paid out after resolving
- resultStorage - result oracle contract address
- prizeCalculator - prize calculator formula contract address

Market functions

- **initialize** - market owner setup market.

- **addPrediction** - owner can add prediction and update in case of issue.
- **changePredictionStatus** - [emergency function] owner can pause prediction when something wrong happened.
- **resolve** - prediction resolving function will be called after oracle knows prediction winning outcome id.
- **payout** - function dedicated for winning payouts to take theyr rewards.
- **refundUser** - [emergency function] - owner can refund forecast to owner in case of issue.
- **refund** - if prediction was canseled users using this function will get refund.
- **receiveApproval** - AIX token will call this function to setup user forecast.
- **transferToPool** - After the product end return leftover AIX tokens to the pool.

View functions are used, because of current solidity limitations:

- **getForecast** - ability to read forecast details
- **getOutcomeTokens** - ability to know each outcome collected money

Safety functions:

- **withdrawETH** - withdraw all ethers in case something wrong will be found
- **withdrawTokens** - withdraw all tokens in case something wrong will be found
- **pause** - pause market in case something wrong will be found

Prize calculator contract

This contract stores formula how much participant can withdraw when his selected output wins.

Function to calculate win amount:

- **calculatePrizeAmount**

At first version we use simples prize distribution formula:

```
Your contributed tokens (_forecastTokens) * _predictionTotalTokens / _
↪winOutputTotalTokens = prize

Example:

A prediction has 2 different outcomes (Outcome 1, Outcome 2) and initial 2000 tokens␣
↪prize:

User A placed 100 tokens on Outcome 1
User B placed 300 tokens on Outcome 1
User C placed 100 tokens on Outcome 2
Total contributed tokens: 2500

After the prediction resolves, the Oracle decides the winning outcome is Outcome 1.
Users can now withdraw the following token amount from their forecasts:

User A -> 100 * 2500 / (100 + 300) = 625 Tokens
User B -> 300 * 2500 / (100 + 300) = 1875 Tokens
User C -> 0 tokens
```

Result Storage contract

This contract store all predictions resolutions. The contract will be connected with our oracle service which will solve predictions. At this moment 3rd party services like http://www.oraclize.it/ are not stable, have limited support, expensive and execution flow is not traceable. When we allow users to create predictions by themselves this part becomes extremely complex and we think that this functionality is a key feature. Having own oracle service will be much flexible solution and in the future, this solution can act as a proxy to better 3rd party oracles.

Platform Part

Platform API:

REST api service for other applications. Users can use to integrate with the platform. Addresses can be found at developers wiki page https://aigangnetwork.github.io/

Platform WEB:

Aigang team maintainable web interface working on top of REST API.

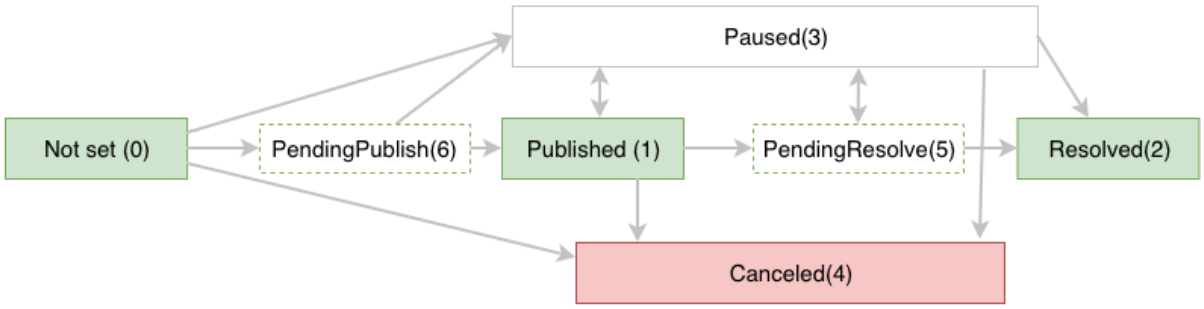
Background services:

Aigang.Predictions.Listener service - helps to maintain prediction statuses. One of jobs example is - older than 24 hourse draft forecasts will be deleted.

Aigang.Transactions.Listener service - helps to maintain blockchain transactions statuses. Example: when forecast receives payment transaction this service activate forecast and update status to “Paid”.

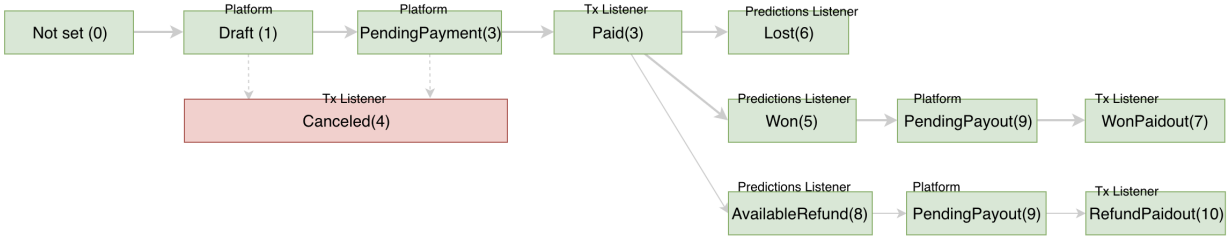
General prediction statuses in platform:

- **PendingPublish (6)** - Prediction is created and waiting for activation.
- **Published (1)** - prediction is ready for market. Initial status.
- **PendingResolve (5)** - Forecasting is ended and waiting for resolving.
- **Resolved (2)** - the prediction winning outcome is known and payouts are ready.
- **Paused (3)** - participation in this predictions is paused and administrators are investigating what is happening
- **Canceled (4)** - some issue happened and refunds for this prediction participants will happen.



General forecast statuses in platform:

- **Draft (1)** - initial forecast status.
- **PendingPayment (2)** - forecast payment was initialized.
- **Paid (3)** - payment was received and forecast is active.
- **Canceled (4)** - forecast is ended.
- **Won (5)** - forecast output index was actual event.
- **Lost (6)** - forecast lost.
- **WonPaidout (7)** - forecast is ended and payout is sent.
- **AvailableRefund (8)** - prediction is canceled and you can refund your tokens.
- **PendingPayout (9)** - payout transactions was submited and platform is waiting until it will be commited.
- **RefundPaidout (10)** - refund was successful and tokens was sent to user.



1.3.3.2 Links

- Prediction Market architecture part 1 (Medium): https://medium.com/aigang-network/prediction-market-architecture-part-1-40756bc358bb
- Prediction Market architecture Part 2 (Medium): https://medium.com/aigang-network/prediction-market-architecture-part-2-a89bf76d59df
- Aigang Platform Contracts Prediction Market (git): https://github.com/AigangNetwork/aigang-platform-contracts-predictions

1.4 Pools

Space for contributions where users can particitpate with theyr own AIX tokens into pool. All collected tokens (pool) will participate in insurance product reserves. After finishing insurance product left reserves return to the pool and will be distributed for pool contributors.

1.4.1 Domain language

- **Pool** - the collected amount of AIX tokens dedicated for the product.
- **Contribution**—your receipt approving participation in the pool.
- **Reward** - amount of AIX tokens returned to the contributor.

1.4.2 Successful contribution flow

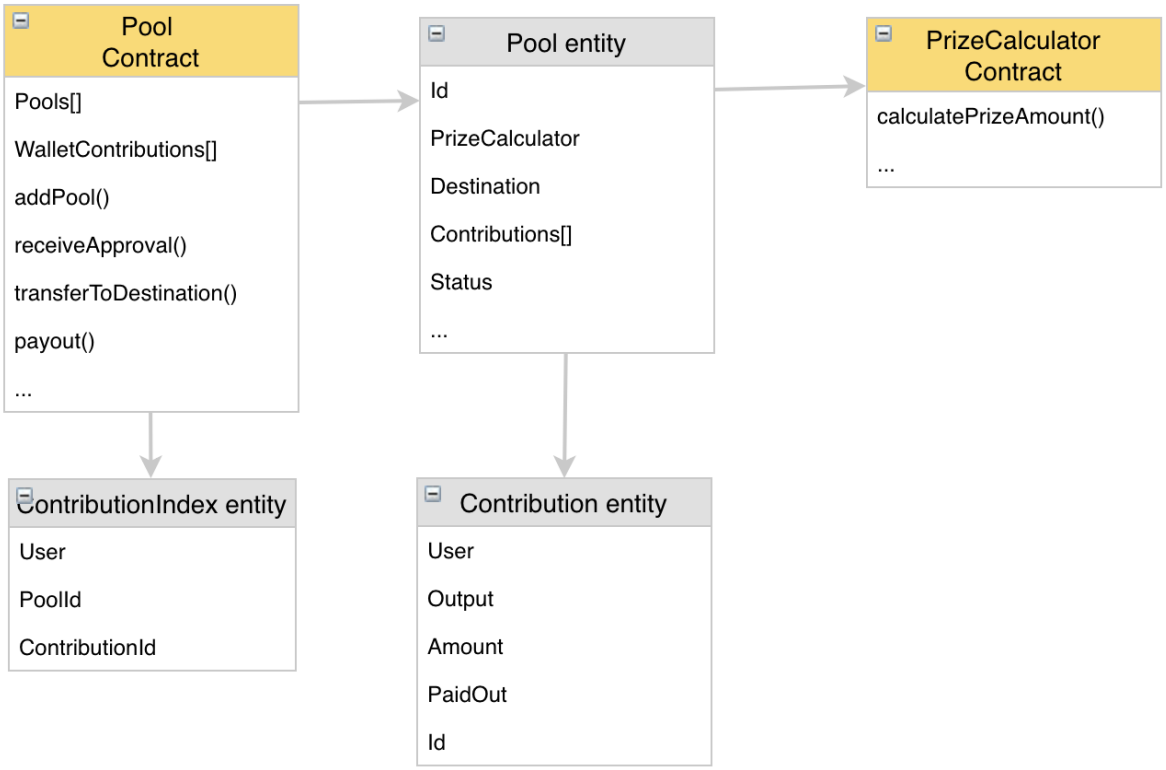
- A user selects an active pool.
- Contribute his amount to the pool.
- Wait until the insurance product ends.
- Take contribution Reward.

1.4.3 More details

1.4.3.1 Pools structure

Blockchain Part

Full view:



Pools Contract

The main contract will be **Pools** contract which holds:

- AIX Balance of all pools
- All actions Events
- Available pools list

- Contributors contribution details

Pool

This entity holds all contributions and address to PrizeCalculator. Full Entity structure:

- id - 32 symbols hash
- contributionStartUtc - (UNIX timestamp) Start date when the pool starts to accept contributions.
- contributionEndUtc - (UNIX timestamp) End date until which the pool accept contributions.
- destination - address of the product to which this pool is dedicated.
- status -
  - NotSet (0)
  - Active (1) - the pool is ready for contributions. Initial status.
  - Distributing (2) - insurance product is ended and rewards are ready.
  - Funded (3) - pool contributions are ended and funds are sent to the insurance product.
  - Paused (4) - the pool is paused by administrators.
  - Canceled (5) - some issue happened and refunds for this pool contributors will happen.
- amountLimit - limit how much tokens pool will collect
- amountCollected - a number of tokens were collected.
- amountDistributing - a number of tokens will be distributed as a reward.
- paidout - a number of tokens were already paid out.
- prizeCalculator - address of reward formula calculator.
- contributions - an array of contributions:
  - id - 32 symbols hash
  - owner - contribution owner wallet address
  - amount - contribution size in AIX tokens
  - paidout - if the pool is canceled or pool distributing - an amount which was transferred for user

Pools functions

- **initialize** - pools owner setup contract
- **addPool** - owner can add pool and update in case of issue
- **setPoolStatus** - owner can change pool status
- **setPoolAmountDistributing** - owner set pool for distributing.
- **receiveApproval** - AIX token will call this function to setup contribution
- **transferToDestination** - transfer pool AIX tokens amount to destination.
- **payout** - function to take contribution rewards
- **refund** - if pool was canseled users using this function will get refund.

View functions are used, because of current solidity limitations

- **getContribution** - ability to read contribution details

Safety functions

- **withdrawETH** - withdraw all ethers in case something wrong will be found
- **withdrawTokens** - withdraw all tokens in case something wrong will be found
- **pause** - pause pools contract in case something wrong will be found

Prize calculator contract

This contract stores formula how much contributor can withdraw:

Function to calculate reward amount:

- **calculatePrizeAmount**

At first version we use simples reward distribution formula:

```
Your contributed tokens (_contributionTokens) * _distributeTotalTokens / _
↪collectedTotalTokens = reward

Example:

A pool collected 2000 tokens reserve:

User A placed 1500 tokens
User B placed 500 tokens

After insurance product ended and return 2800 tokens as leftover to pool.
Users can now withdraw the following token amount from their contribution:

User A -> 1500 * 2800 / 2000 = 2100 Tokens
User B -> 500 * 2800 / 2000 = 700 Tokens
```

Platform Part

Platform API:

REST api service for other applications. Users can use to integrate with the platform. Addresses can be found at developers wiki page https://aigangnetwork.github.io/

Platform WEB:

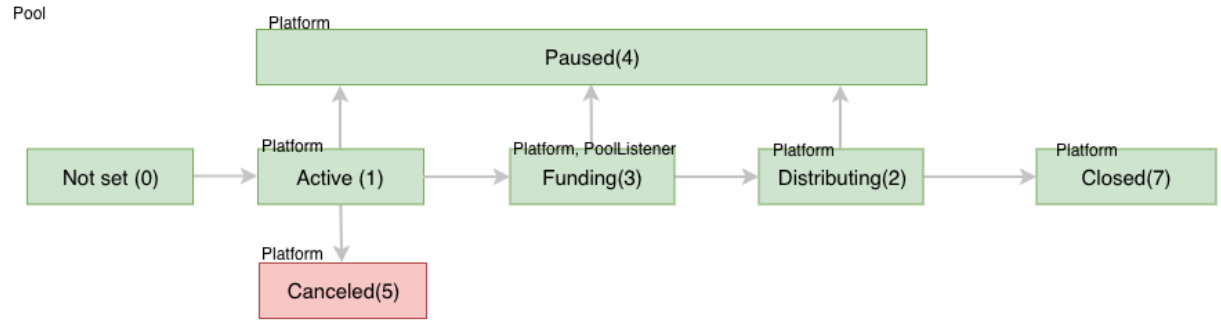
Aigang team maintainable web interface working on top of REST API.

Background services:

Aigang.Transactions.Listener service - helps to maintain blockchain transactions statuses. Example: when contribution receives payment transaction this service activate contribution and update status to "Paid". **Aigang.Pools.Listener** service - helps to maintain pools and contributions. Example: when contribution doesn't have payment transaction after 1-hour, status will be changed to "Canceled".

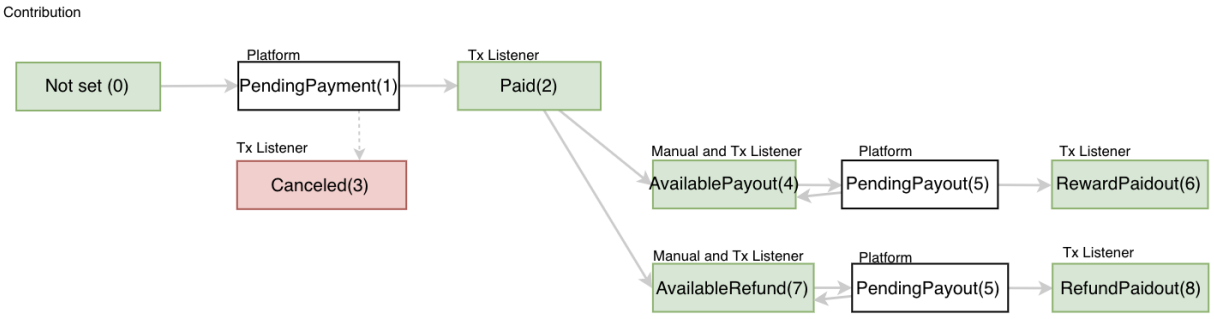
General pool statuses in platform:

- **Active (1)** - pool is ready to accept contributions. Initial status.
- **Distributing (2)** - the pool is distributing rewards.
- **Funding (3)** - Contributing is ended and collected pool will be sent to destination.
- **Paused (4)** - contribution in this pool is paused and administrators are investigating what is happening
- **Canceled (5)** - some issue happened and refunds for this pool contributors will happen.
- **Closed (7)** - final state when all rewards are taken.



General contribution statuses in platform:

- **PendingPayment (1)** - contribution payment was initialized.
- **Paid (2)** - contribution was received.
- **Canceled (3)** - contribution is canceled because paiment was not received in 24 hours.
- **AvailablePayout (4)** - reward is available to you to take out.
- **AvailableRefund (7)** - pool is canceled and you can refund your tokens.
- **RewardPaidout (6)** - reward was successful and tokens was sent to a user.
- **PendingPayout (5)** - payout transactions were submited and the platform is waiting until it will be commited.
- **RefundPaidout (8)** - refund was successful and tokens was sent to a user.



1.4.3.2 Links

- Aigang Platform Contracts Pools (git): https://github.com/AigangNetwork/aigang-platform-contracts-pools
- Pools architecture (Medium): https://medium.com/aigang-network/pools-architecture-a41a9761418c

1.5 Insurance

Space for insurance products where a user can choose insurance products.

Insurance products have two main parts: - Management part: insurance product configuration and description - Financial Blockchain part: auditable contracts with money movements and policies details.

1.5.1 Successful insurance flow

- A user selects insurance product.
- Read the details, terms, conditions, contract about the product and if everything acceptable continues further.
- Follow device pairing instructions.
- Our services will check device properties, compare them with contract rules and if the device is valid DRAFT policy with calculated price will be created.
- If the policy details are acceptable User can pay and activate the policy, or delete if it is not acceptable.

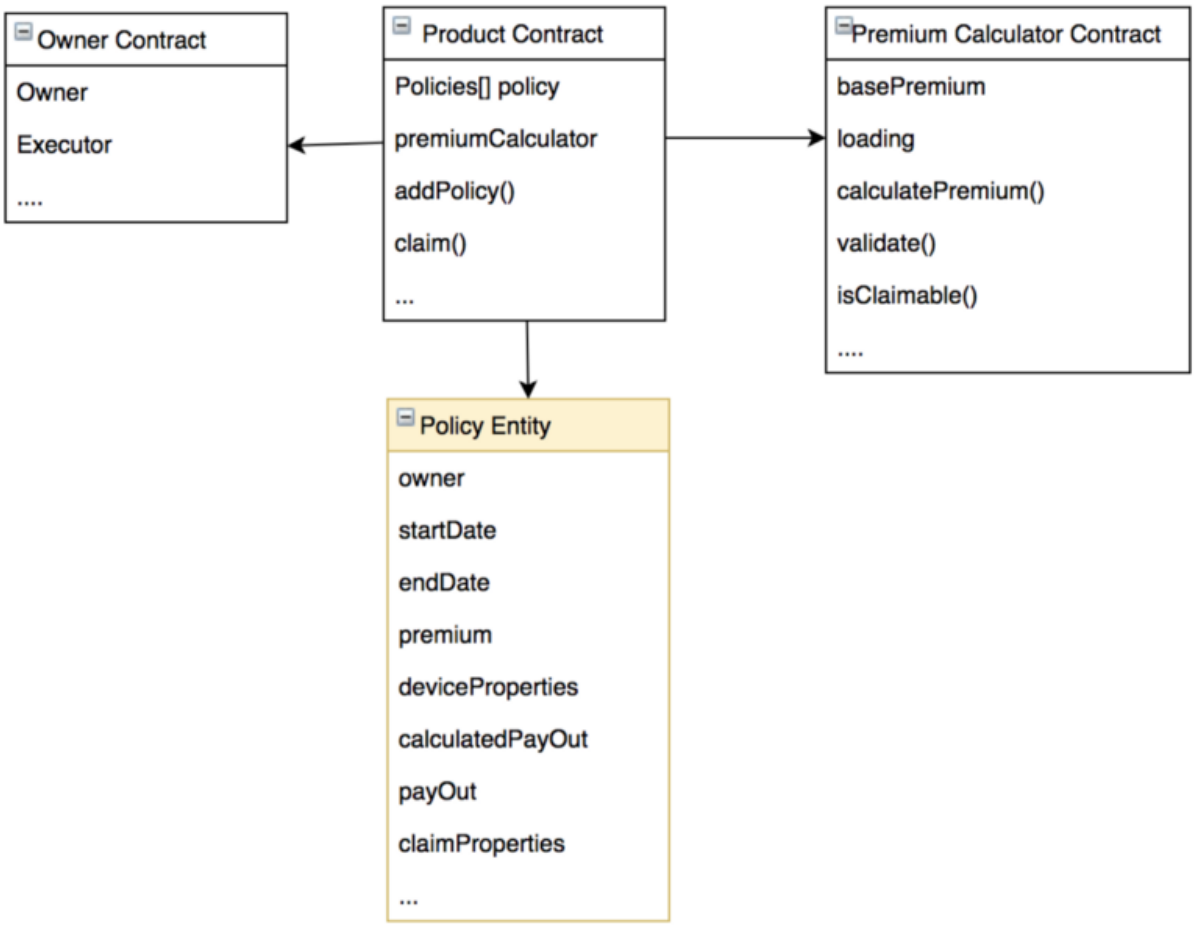
1.5.2 Successful claim flow

- A user selects active policy.
- Follow device claim instructions
- Our services will check device properties, compare them with contract rules and if the device is claimable change policy status to Claimable
- User Claims his payout and payout will be withdrawn to his wallet.

1.5.3 More details

1.5.3.1 Insurance structure

Blockchain Part



The main contract will be **Product** contract which holds all insurance product properties, balance, list of policies and reference to the Premium calculator contract.

Policies contain all audit-able details about policy evaluation dates, premium, payout size, device details on policy writing date and all details about the payout.

Premium Calculator contract will hold a risk model with calculation formulas, base premium, fees and claim validation rules.

Platform Part

Platform API:

REST api service for other applications. Users can use to integrate with the platform. Addresses can be found at developers wiki page https://aigangnetwork.github.io/

Platform WEB:

Aigang team maintainable web interface working on top of REST API.

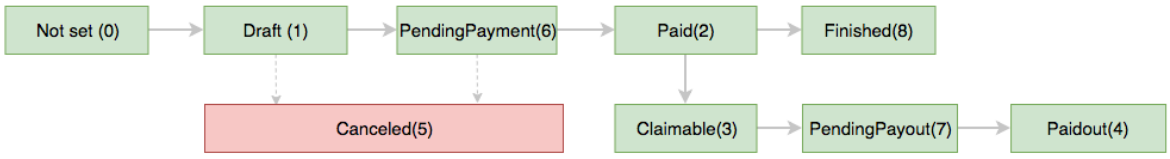
Background services:

Aigang.Policies.Listener service - helps to maintain policies. One of jobs example is - old draft poliecies are moved to status “Canceled” and finished to status “Finished”.

Aigang.Transactions.Listener service - helps to maintain blockchain transactions statuses. Example: when insurance product receives new policy payment transaction this service activate policy and submit it to the Blockchain.

General policy statuses:

- **Draft** - initial policy status.
- **PendingPayment** - policy payment was initialized.
- **Paid** - payment was received and policy is active.
- **Finished** - policy is ended.
- **Claimable** - device was broken and user can take payout.
- **PendingPayout** - payout is initialized.
- **Paidout** - policy is ended and payout is send.
- **Canceled** - for some reasons policy is canceled.



1.5.3.2 Android Battery Insurance Product

This insurance product is working example based on “Model Product for Phone Battery Insurance” data model. Insurance product contains a description, terms and conditions and insurance product contract address.

Calculating Premium

Each user can calculate the specific premium for their device by following these steps:

1. Downloading Aigang App
2. Open the app and pair your device with the Aigang Platform.
3. In the Aigang Platform, press the button “Calculate Premium” button and enter your device ID.

Data from your device will be collected and sent to the premium calculator smart contract. The premium calculator will validate the data and calculate the premium based on your device’s data.

Insuring your device

After calculating your premium you will be able to insure your device. You have to be logged in to your Metamask wallet and have the required amount on AIX tokens. To insure your device just press "Insure" button in the policy window. Metamask will ask you to confirm your transaction. After your transaction is confirmed, your policy status will be changed to "Paid" and your device will be insured.

Insurance Product Smart Contract

All policies are stored in insurance product smart contract. Your policy ID is the key to getting your policy data from the smart contract.

1.5.3.3 Test app install instructions

This section contains documentation how to set up environment for android app testing.

> **Attention:** Aigang android insurance product only allows to use real android devices with SIM card(s).

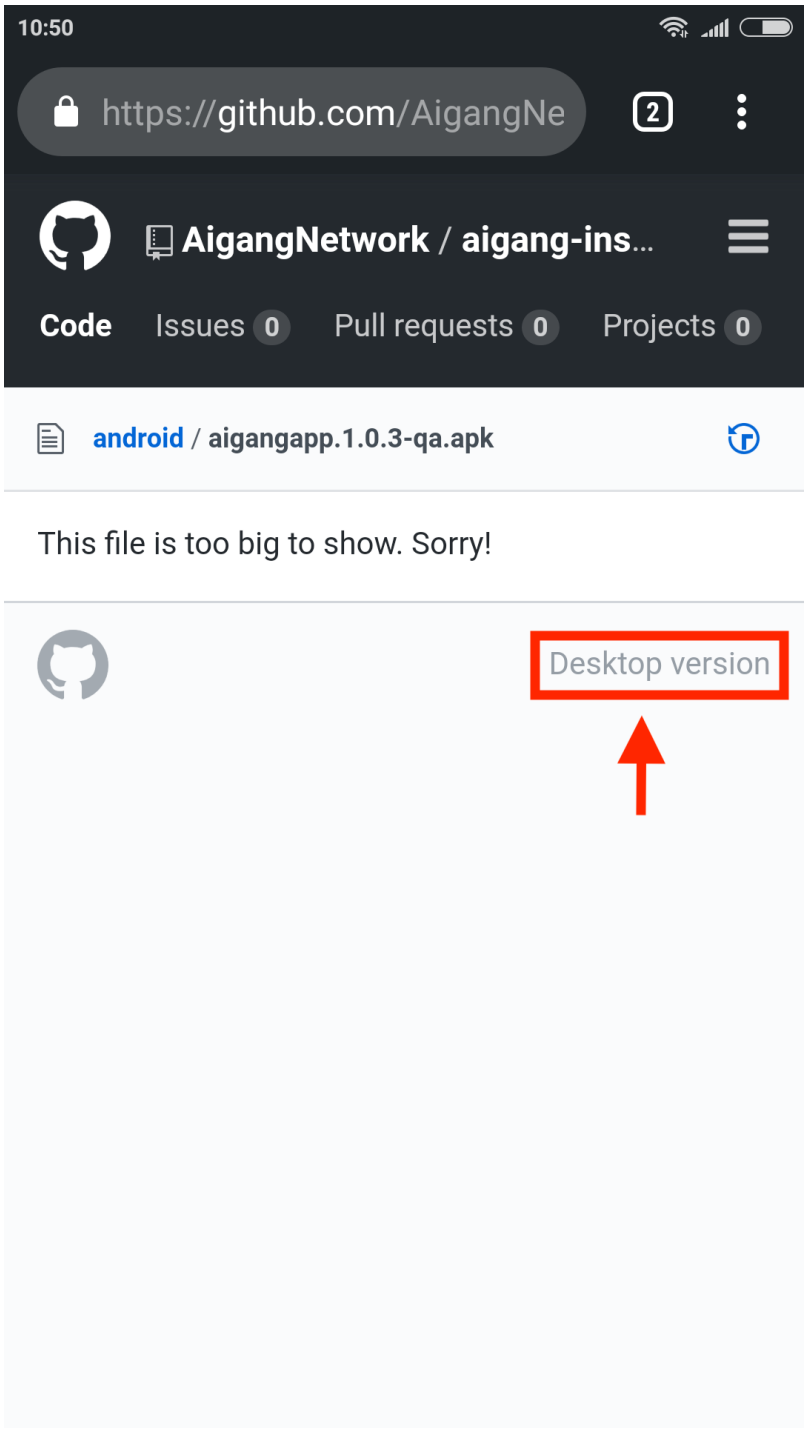
Install test app for Android device

Step 1

- From your android device navigate to https://github.com/AigangNetwork/aigang-insurance-app/tree/master/android
- If there is more than one APK version, choose latest one.

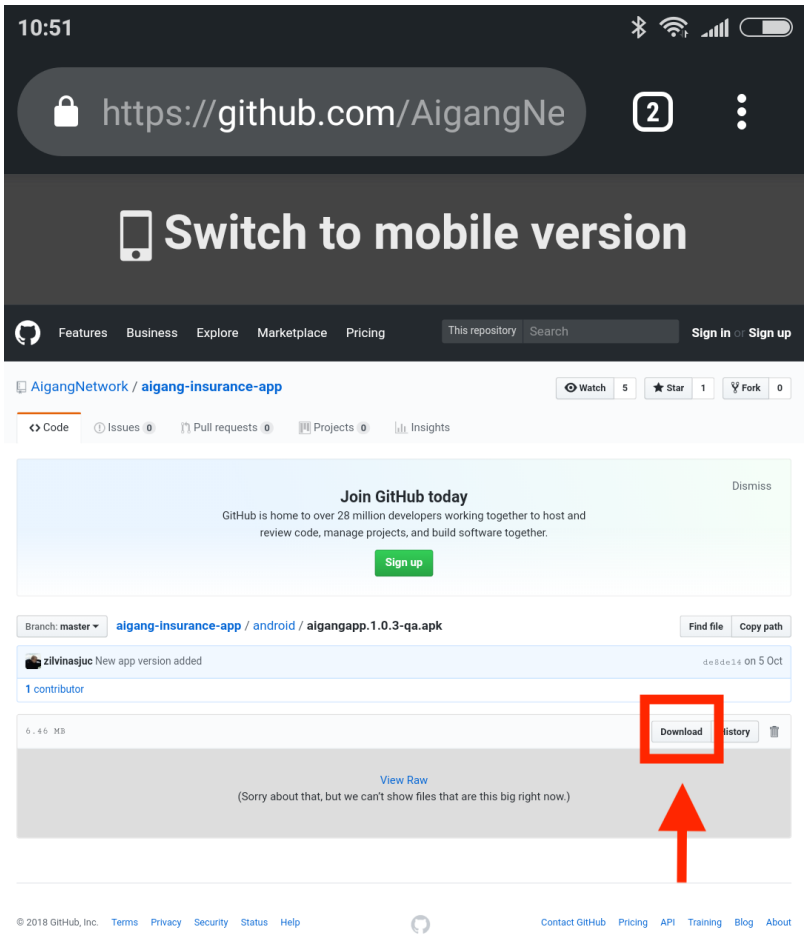
Step 2

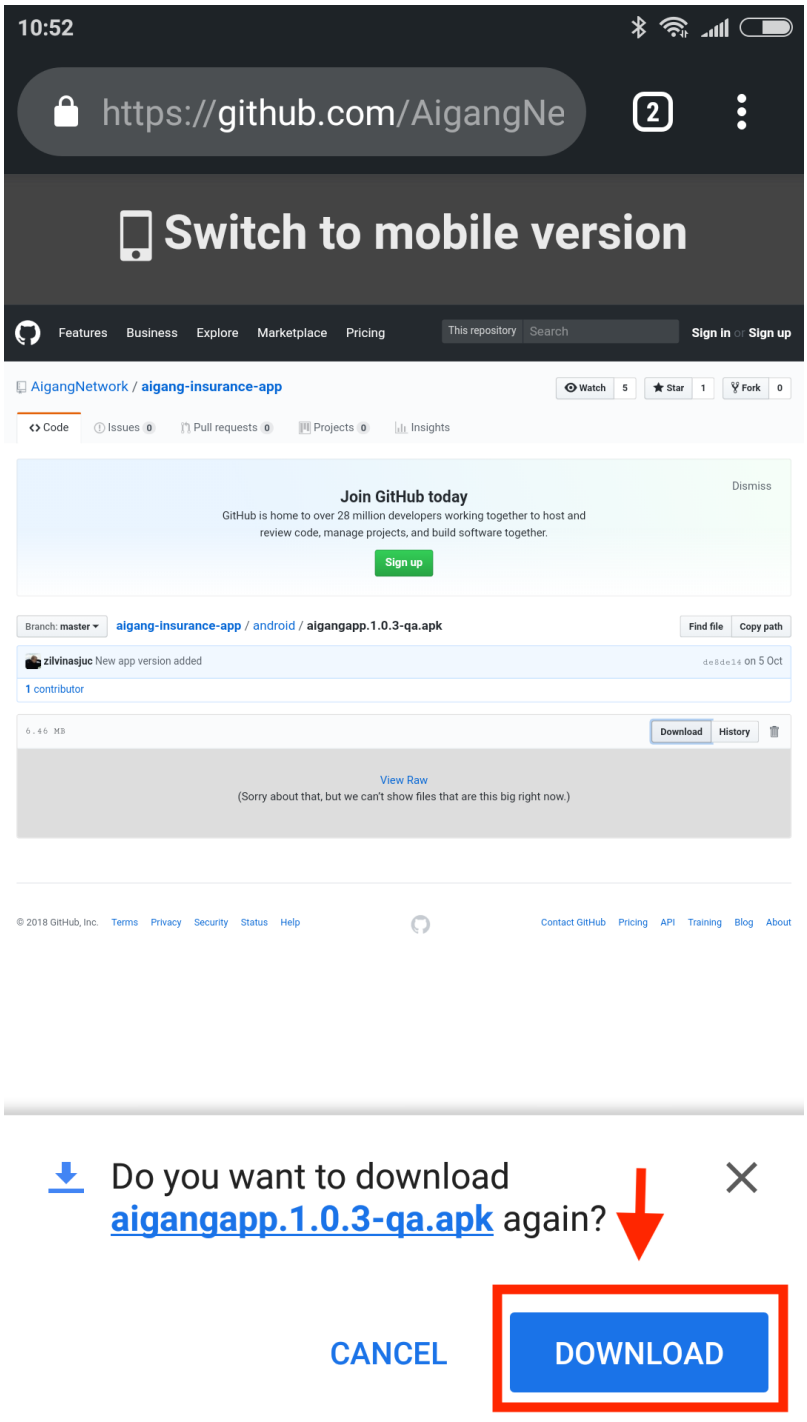
- Switch your mobile browser mode to view desktop mode. It allows you to see Download button.



Step 3

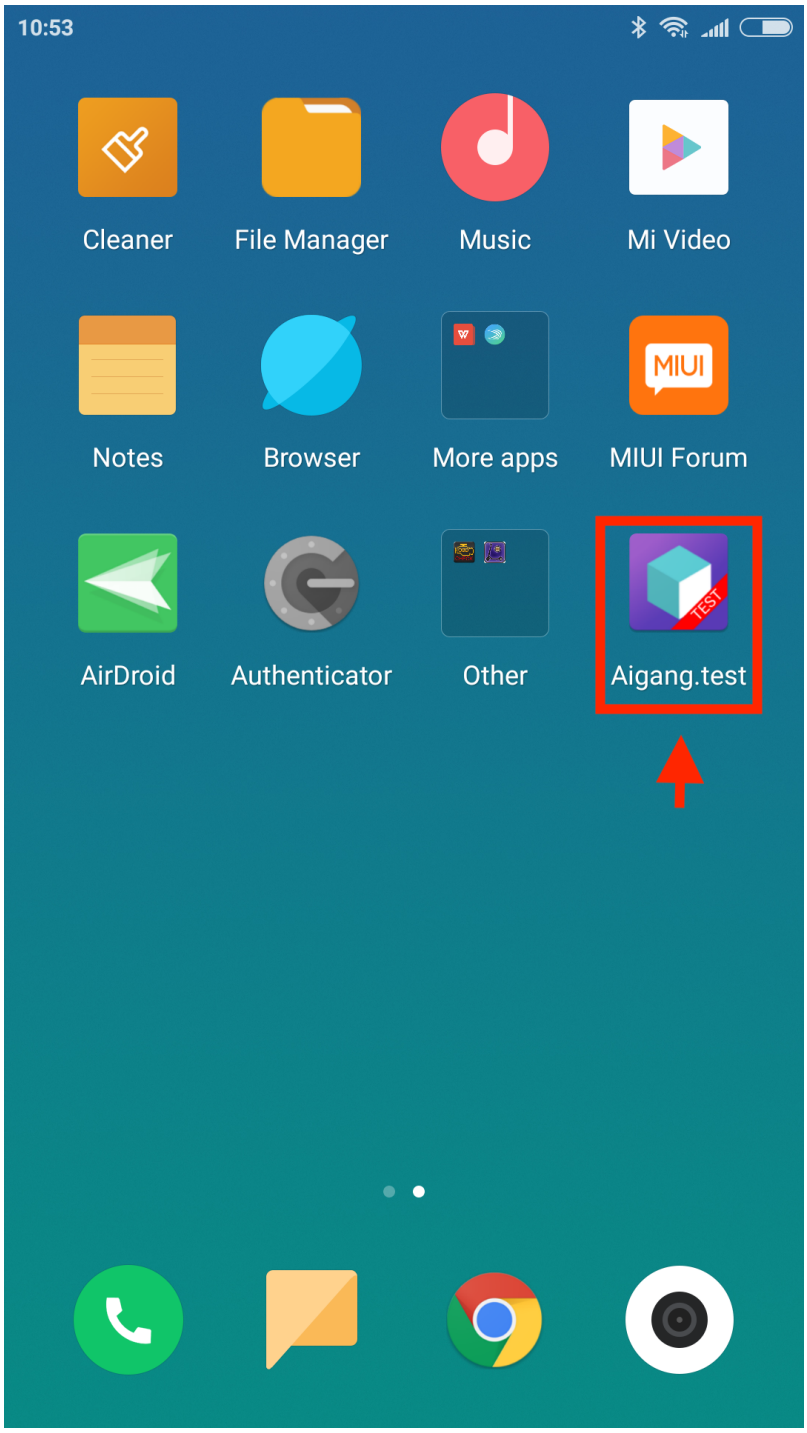
- Click *Download* button and download APK.


10:51
https://github.com/AigangNe
2
Switch to mobile version
Features
Business
Explore
Marketplace
Pricing
This repository
Search
Sign in or Sign up
AigangNetwork / aigang-insurance-app
Watch 5
Star 1
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Insights
Dismiss
Join GitHub today
GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.
Sign up
Branch: master
aigang-insurance-app / android / aigangapp.1.0.3-qa.apk
Find file
Copy path
zilvinasjuc New app version added
de8de14 on 5 Oct
1 contributor
6.46 MB
Download
History
View Raw
(Sorry about that, but we can't show files that are this big right now.)
© 2018 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About



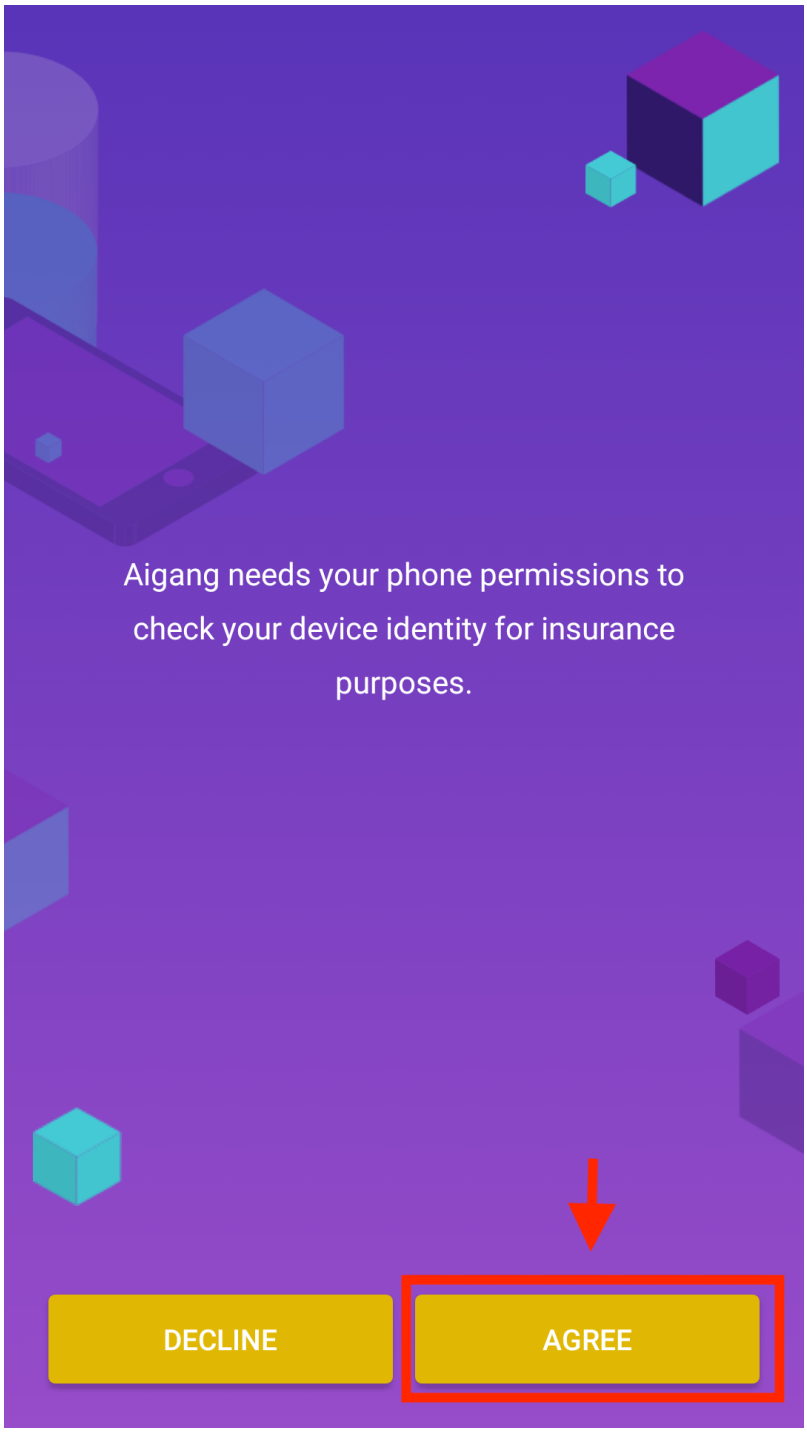
Step 4

- Install and open app.



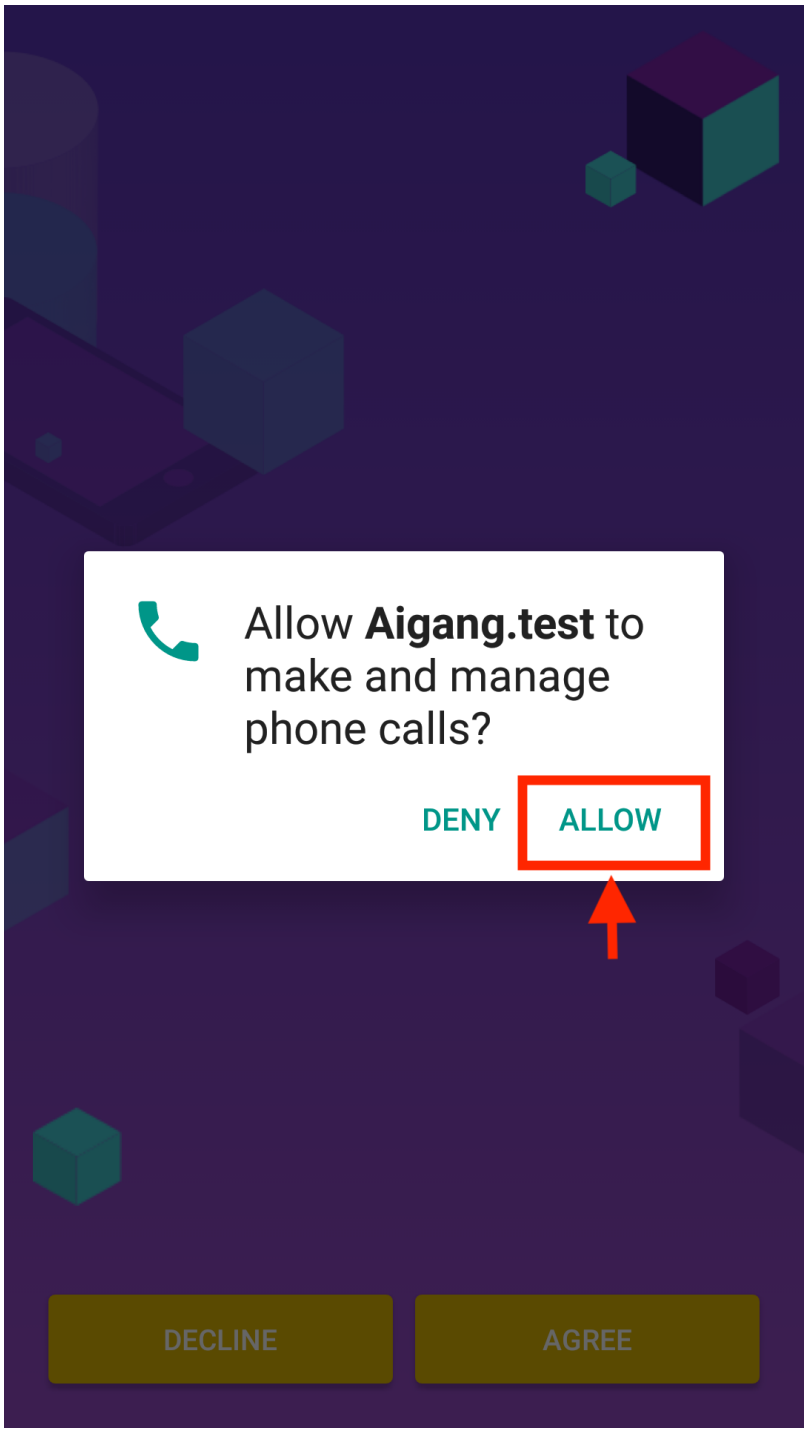
Step 5

- Agree with permission request.



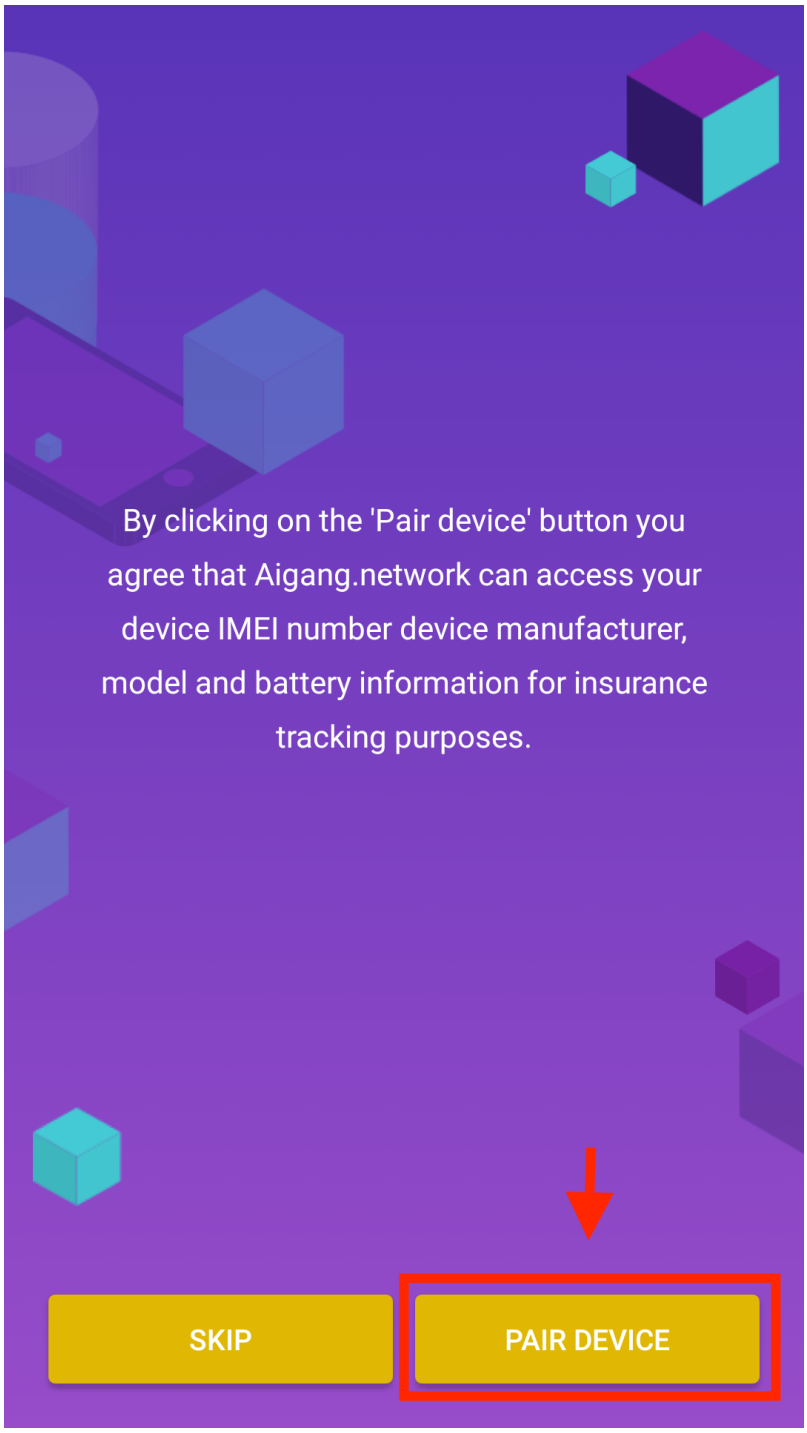
Step 6

- Grant telephony permission for aigang app. Application needs this permission to get SIM card related data.



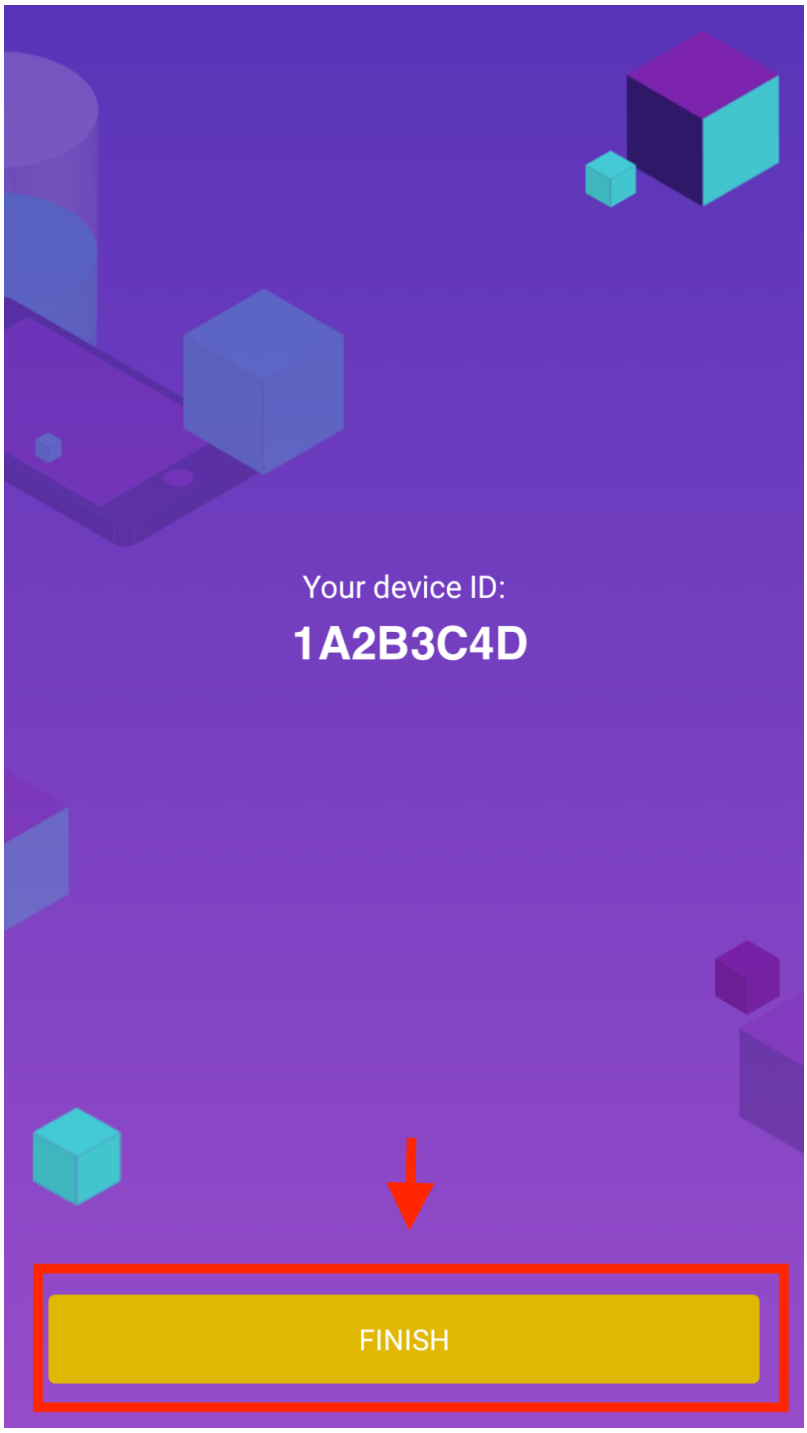
Step 7

- Pair your device with test platform.



Step 8

- Congratulations, you successfully paired your device.



Android application Mock data

In case of testing you can mock real data of your device.

White list

To get ability to insure more then one time whitelist your device ID: https://github.com/AigangNetwork/aigang-platform-web/issues/86

Create test account

- Open https://testplatform.aigang.network/register and sign up.
- Active your test account by clicking link in registration email.

Generate test AIX tokens

- Connect to your wallet client (e.x Metamask)
- Navigate to https://aigangnetwork.github.io/testaix/
- Click *Generate tokens for me*

1.5.3.4 Links

- Battery insurance product (Medium): https://medium.com/aigang-network/battery-insurance-product-f19cab2f35a3
- Aigang Platform Contracts Insurance (git): https://github.com/AigangNetwork/aigang-platform-contracts-insurance

1.6 Documenation examples

Here lays are some examples for writing well-formated documentation

This documentation was written using reStructedText pattern. All concepts and syntax could be found here.

1.6.1 Code

This is a typical paragraph. An indented literal block follows.

```
for a in [5,4,3,2,1]:   # this is program code, shown as-is
    print a
print "it's..."
# a literal block continues until the indentation ends
```

This text has returned to the indentation of the first paragraph, is outside of the literal block, and is therefore treated as an ordinary paragraph.

1.6.2 Tables

row 1, col 1	column 2	column 3	column 4
row 2	Use the	tables	more.
row 3			

1.6.3 Images

1.6.4 Hyperlinks

This is simple link: http://example.com/ This is a paragraph that contains a link.

1.6.5 Lists and emphasis

- one asterisk: *text* for emphasis (italics),
- two asterisks: **text** for strong emphasis (boldface), and
- backquotes: `text` for code samples.

1.6.6 test Paragraph

1.6.6.1 Test1

Test2

Test3

Test4