FINS User Interface

Firmware IP Node Specification

by Lucy

OVERVIEW

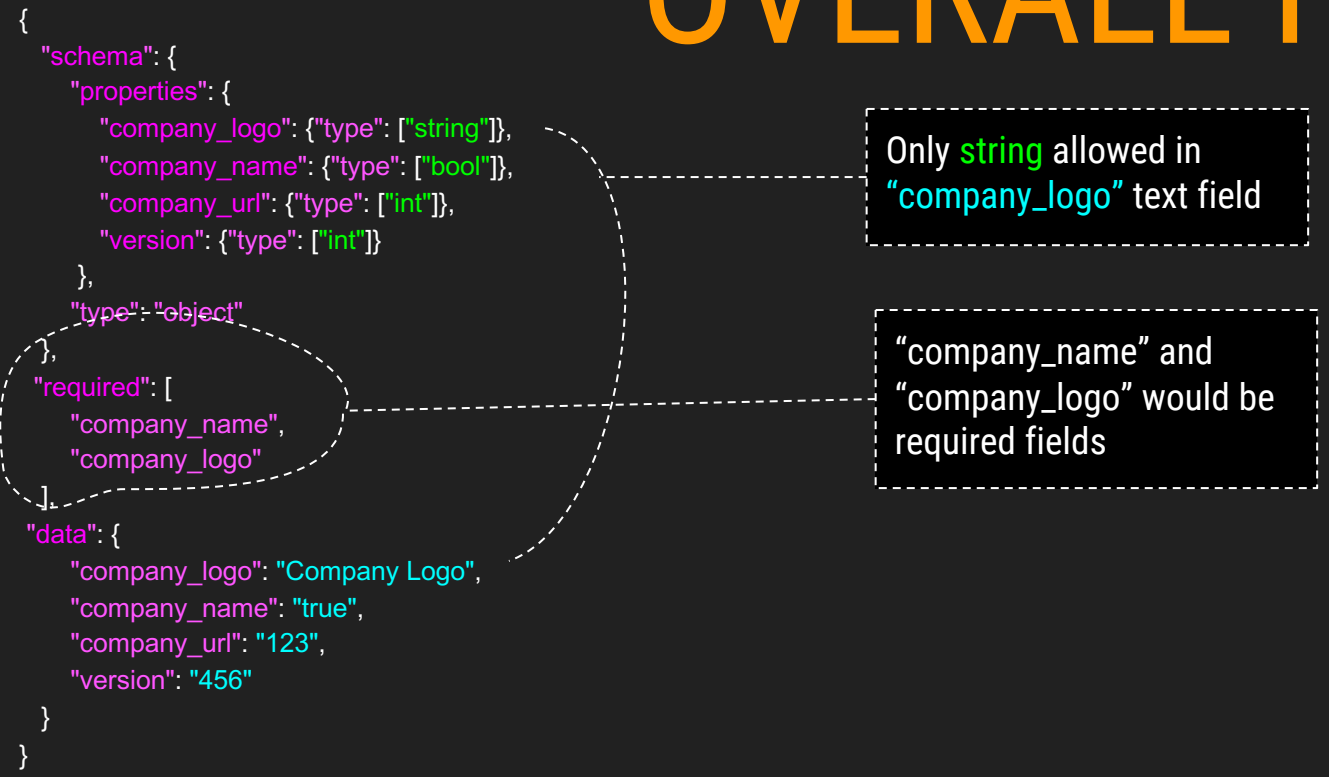
- Goal of the project
- Timeline
- Integrating C++ and QML
- Current UI
- Next steps

OVERALL PROJECT GOAL

- Ability to update a JSON file using UI
  - Using Qt to create the UI with a C++ backend
  - QML → C++ → Update JSON
- Angular JSON Schema Form

"**Qt** is a cross-platform application development framework for desktop, embedded and mobile."
https://wiki.qt.io/About_Qt

OVERALL PROJECT GOAL



Example of a JSON file: An “array” of defining data

TIMELINE

1. Getting Familiar with Qt

- What is it
- Viewing examples
- Signals and slots

2. Creating Example Forms

- Based off of Angular JSON Schema
- Incorporating JSON with UI using Qt's UI creator

3. Incorporating C++

- Reading/writing JSON file using C++
- Using signals and slots
- Hard-code vs dynamically

INTEGRATING C++ AND QML TO CONNECT TO JSON

C++ opens and reads JSON file

SETTING INITIAL VARIABLES

Update JSON when text field is edited

SIGNALS, Q_PROPERTY, AND OTHER FUN FUNCTIONS

main.cp

```
// OPENING JSON FILE
    QFile file("C:/Users/lappiah/Documents/PropertyTest/fins-schema.json");
    if (!file.exists()) qDebug() << "NO FILE FOUND";
    file.open(QIODevice::ReadOnly);
    QByteArray rawData = file.readAll();
    QJsonDocument doc(QJsonDocument::fromJson(rawData));
    QJsonObject root = doc.object();
```

main.cp

```
// SETTING UP VARIABLES
    QJsonValue jv2 = root.value("data");

    QJsonValue schemaVal = root.value("schema");
    QJsonObject schemaTree = schemaVal.toObject();
    QJsonValue properties = schemaTree.value("properties");
    QJsonObject properTrees = properties.toObject();
    QJsonValue reqFieldsVal = root.value("required");
    QJsonArray reqFieldsArr = reqFieldsVal.toArray();
```

backend.h

```
    Q_PROPERTY(QString name READ name WRITE setName NOTIFY nameChanged)
    Q_PROPERTY(QString companyLogo READ companyLogo WRITE setCompanyLogo NOTIFY companyLogoChanged)
    Q_PROPERTY(QString companyURL READ companyURL WRITE setCompanyURL NOTIFY companyURLChanged)
    Q_PROPERTY(QString description READ description WRITE setDescription NOTIFY descriptionChanged)
    Q_PROPERTY(QString modelingTool READ modelingTool WRITE setModelingTool NOTIFY
modelingToolChanged)
    Q_PROPERTY(QString companyName READ companyName WRITE setCompanyName NOTIFY companyNameChanged)
    . . .
```

INTEGRATING C++ AND QML TO CONNECT TO JSON

SIGNALS, Q_PROPERTY, AND OTHER FUN FUNCTIONS

```
Q_PROPERTY(QString description READ description WRITE setDescription NOTIFY descriptionChanged)
```

Q_PROPERTY RELATED FUNCTIONS

```
explicit BackEnd(QObject *parent = nullptr);

QString description();

void setDescription(const QString &description);

void descriptionChanged ();

QString m_description;
```

← SIGNAL

OTHER FUNCTIONS

```
QByteArray startRead();
void writeFile(QJsonDocument doc);
void insertData(QString where, QString what);
void appendData(QString what);

Q_INVOKABLE void staySameFile();
Q_INVOKABLE void newBackUpFile();
Q_INVOKABLE void delEntry();
```

Backend Functions

INTEGRATING C++ AND QML TO CONNECT TO JSON

SIGNALS, Q_PROPERTY, AND OTHER FUN FUNCTIONS

Update m_description variable

QML

START HERE!

```
QString m_description;
```

```
QString BackEnd::description() {
    return m_description;
}
```

Update JSON

```
void BackEnd::setDescription(const QString &description)
{
    if(description == m_description)
        return;
    m_description = description;
    insertData("description", m_description);
    emit descriptionChanged();
}
```

```
void BackEnd::insertData(QString where, QString what) {
    QJsonDocument doc(QJsonDocument::fromJson(startRead()));
    QJsonObject root = doc.object();
    QJsonValueRef ref = root.find("data").value();
    QJsonObject m_addvalue = ref.toObject();
    m_addvalue.insert(where, what);
    ref = m_addvalue;
    doc.setObject(root);
    writeFile(doc);
}
```

INTEGRATING C++ AND QML TO CONNECT TO JSON

SIGNALS, Q_PROPERTY, AND OTHER FUN FUNCTIONS

MAIN.QML

Creating an instance of BackEnd

```
BackEnd {
    id: backend
}
```

```
TextFieldInputReq {
    text: backend.description
    objectName: "description"
    onTextChanged: backend.description =
text
}
```

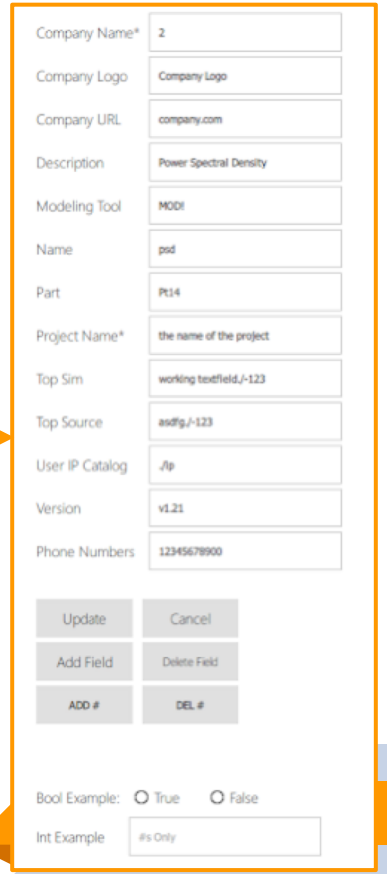
CURRENT USER INTERFACE


```
{
  "data": {
    "company_logo": "Company Logo",
    "company_name": "2",
    "company_url": "company.com",
    "description": "Power Spectral Density",
    "modeling_tool": "MOD!",
    "name": "psd",
    "part": "Pt14",
    "phone_numbers": [
      {
        "number": "12345678900"
      }
    ],
    "project_name": "the name of the project",
    "top_sim": "working textfield./-123",
    "top_source": "asdfg./-123",
    "user_ip_catalog": "./ip",
    "version": "v1.21"
  },
  "required": [
    "company_name",
    "project_name"
  ],
   . . .
```

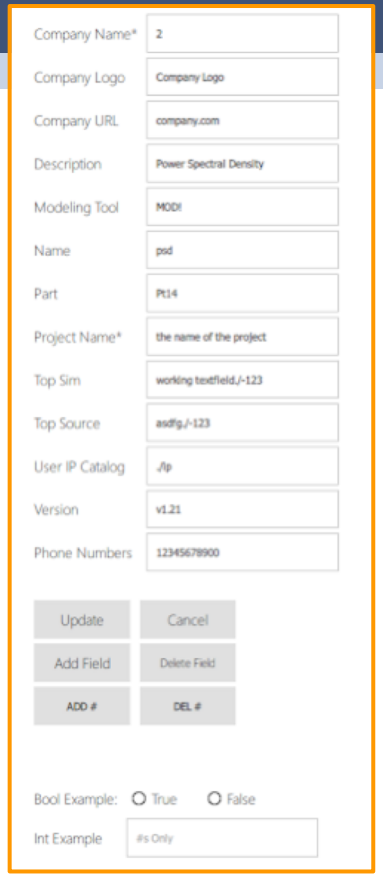

```
"schema": {
  "properties": {
    "company_logo": {
      "type": [
        "string"
      ]
    },
```

Using C++, JSON, and QML to populate UI

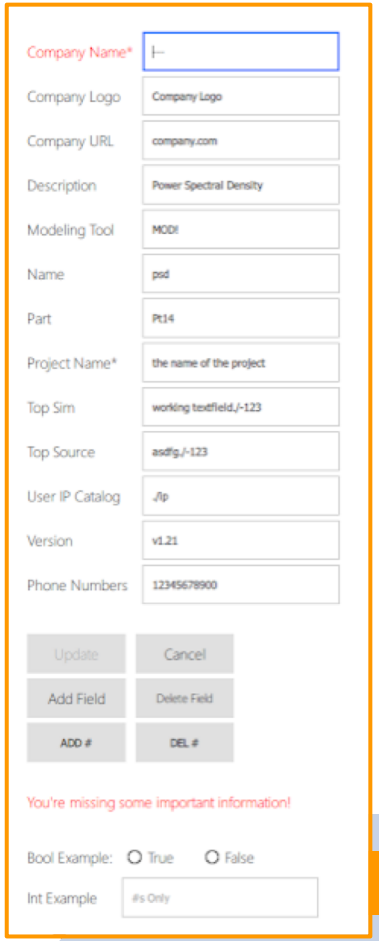
(PREVIOUS SLIDES)



CURRENT USER INTERFACE



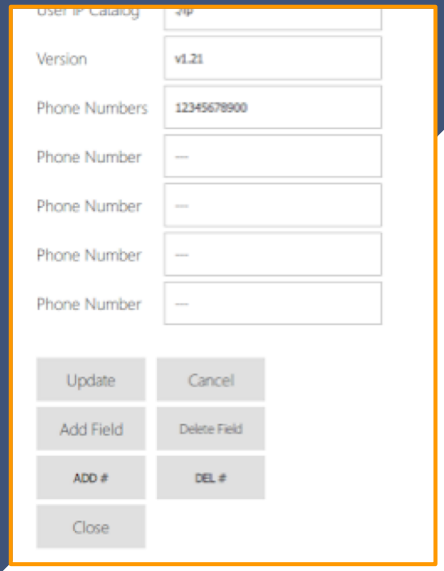
- Updates JSON when edited
- Required fields:
  - Highlights fields
  - Unable to click update button
- Ability to create acceptable input using RegEx
  - Phone Numbers can only be numbers
  - 0 - 15 numbers allowed
- Dynamically creates new fields



NEXT STEPS. . .

WORKING ON NOW

- ❑ Dynamically creating fields in UI (using QML)
  - ❑ ~~"Add #" clicked → create a phone number new field~~
  - ❑ "Del #" clicked → delete a phone number field
- ❑ Connect dynamically created QML objects to instance of BackEnd class
  - ❑ ~~Must have access to functions (description ex.)~~
  - ❑ Correctly append to JSON file → out of order & duplicates
  - ❑ ~~Creating initial fields at runtime depending on JSON~~
  - ❑ Create multiple fields at a time (Angular, phone# example)
- ❑ Fix bugs



THANKS!

Any questions?