How to Add KPIs to the Simulator

Step 1: Determine what Data-type your KPI has to be

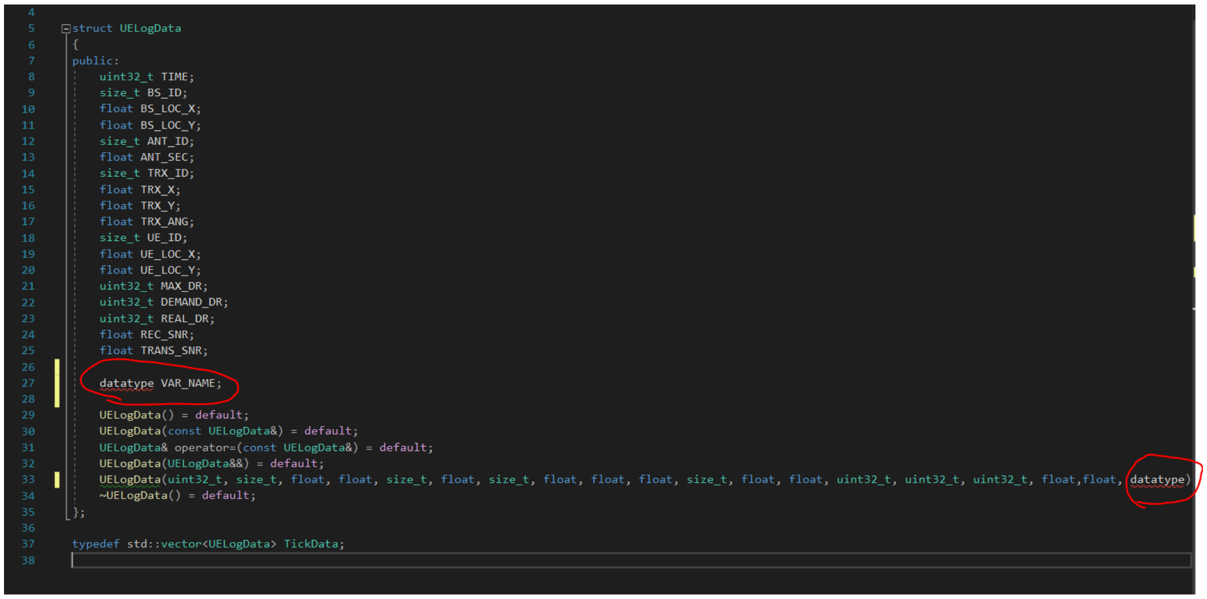
most likely it will either be an uint32_t or a float

Step 2: Add your variable to UELogData.h and UELogData.cpp

In UELogData.h:

Add it onto the list of public variables

Also add it as a parameter in the function declaration for UELogData::UELogData()



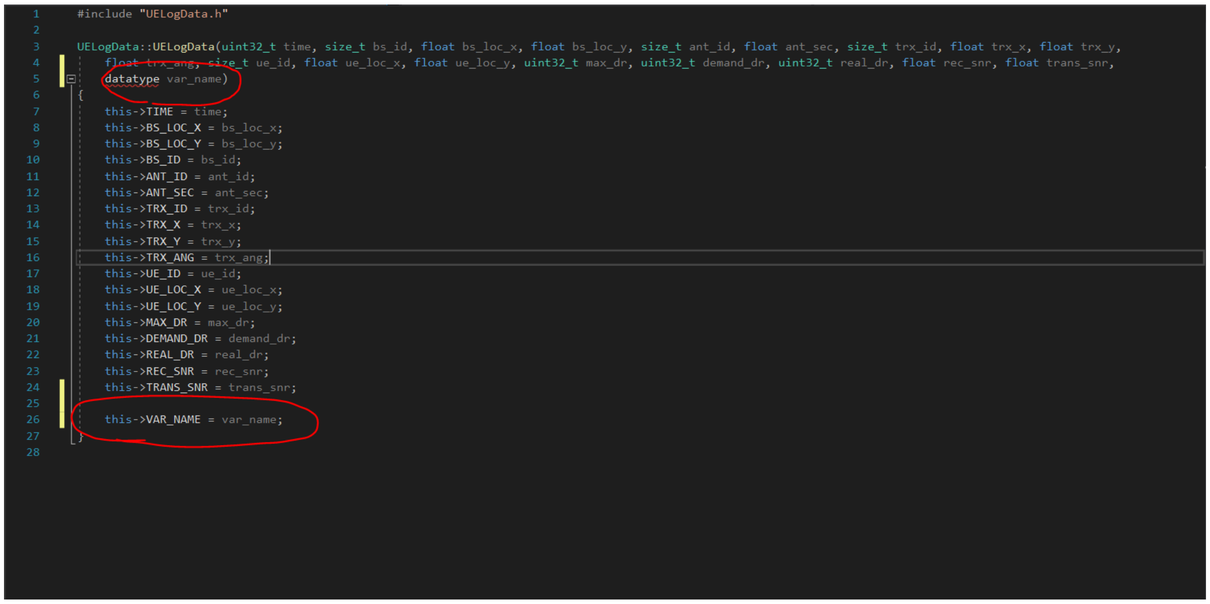
```
struct UELogData
{
public:
    uint32_t TIME;
    size_t BS_ID;
    float BS_LOC_X;
    float BS_LOC_Y;
    size_t ANT_ID;
    float ANT_SEC;
    size_t TRX_ID;
    float TRX_X;
    float TRX_Y;
    float TRX_ANG;
    size_t UE_ID;
    float UE_LOC_X;
    float UE_LOC_Y;
    uint32_t MAX_DR;
    uint32_t DEMAND_DR;
    uint32_t REAL_DR;
    float REC_SNR;
    float TRANS_SNR;

    datatype VAR_NAME;

    UELogData() = default;
    UELogData(const UELogData&) = default;
    UELogData& operator=(const UELogData&) = default;
    UELogData(UELogData&&) = default;
    UELogData(uint32_t, size_t, float, float, size_t, float, size_t, float, float, float, size_t, float, float, uint32_t, uint32_t, uint32_t, float,float, datatype)
    ~UELogData() = default;
};

typedef std::vector<UELogData> TickData;
```

In UELogData.cpp

Add it to the function parameters of UELogData::UELogData()

Add it into the function by writing this line at the bottom of the function:

this->VAR_NAME = var_name;



```cpp
#include "UELogData.h"

UELogData::UELogData(uint32_t time, size_t bs_id, float bs_loc_x, float bs_loc_y, size_t ant_id, float ant_sec, size_t trx_id, float trx_x, float trx_y,
    float trx_ang, size_t ue_id, float ue_loc_x, float ue_loc_y, uint32_t max_dr, uint32_t demand_dr, uint32_t real_dr, float rec_snr, float trans_snr,
    datatype var_name)
{
    this->TIME = time;
    this->BS_LOC_X = bs_loc_x;
    this->BS_LOC_Y = bs_loc_y;
    this->BS_ID = bs_id;
    this->ANT_ID = ant_id;
    this->ANT_SEC = ant_sec;
    this->TRX_ID = trx_id;
    this->TRX_X = trx_x;
    this->TRX_Y = trx_y;
    this->TRX_ANG = trx_ang;
    this->UE_ID = ue_id;
    this->UE_LOC_X = ue_loc_x;
    this->UE_LOC_Y = ue_loc_y;
    this->MAX_DR = max_dr;
    this->DEMAND_DR = demand_dr;
    this->REAL_DR = real_dr;
    this->REC_SNR = rec_snr;
    this->TRANS_SNR = trans_snr;

    this->VAR_NAME = var_name;
}

```

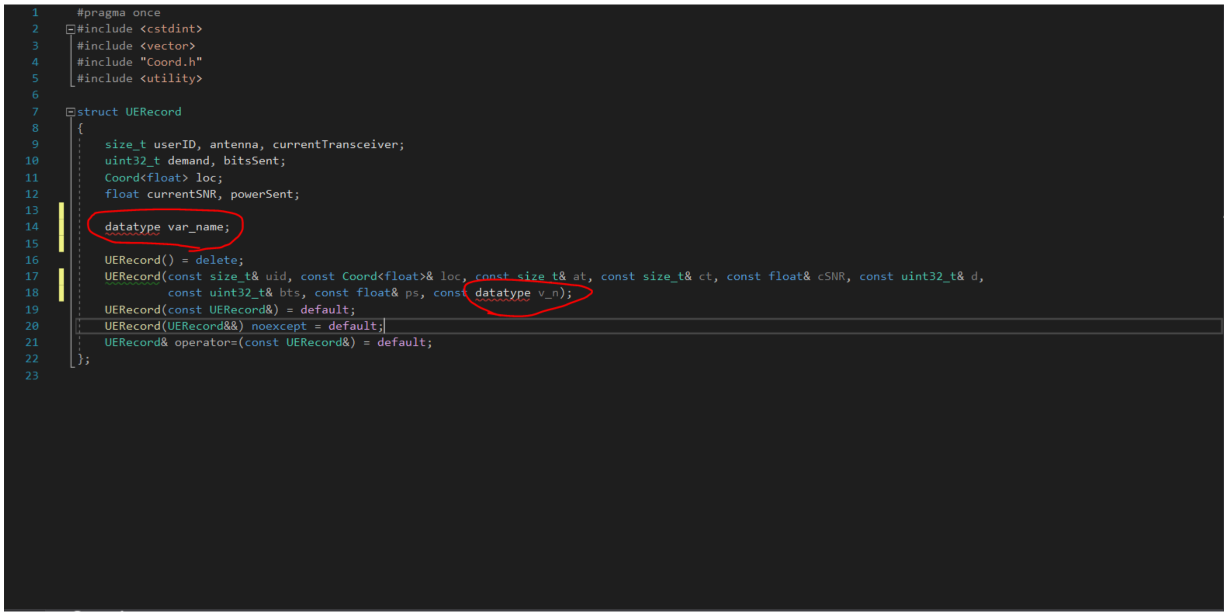
Note: replace VAR_NAME with the actual name you want to use for the variable

Step 3: Add your variable to UERecord.h and UERecord.cpp

In UERecord.h:

Add the variable to the list of declared variables

Also add it as a parameter in the function declaration for UERecord::UERecord()



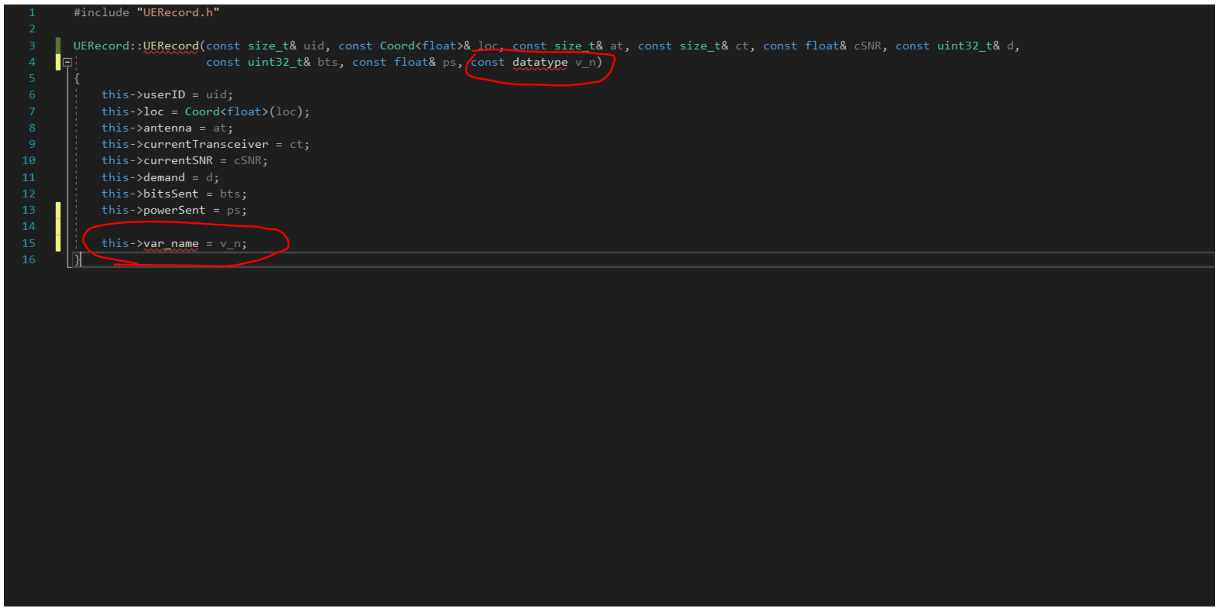
```cpp
#pragma once
#include <cstdint>
#include <vector>
#include "Coord.h"
#include <utility>

struct UERecord
{
    size_t userID, antenna, currentTransceiver;
    uint32_t demand, bitsSent;
    Coord<float> loc;
    float currentSNR, powerSent;

    datatype var_name;

    UERecord() = delete;
    UERecord(const size_t& uid, const Coord<float>& loc, const size_t& at, const size_t& ct, const float& cSNR, const uint32_t& d,
             const uint32_t& bts, const float& ps, const datatype v_n);
    UERecord(const UERecord&) = default;
    UERecord(UERecord&&) noexcept = default;
    UERecord& operator=(const UERecord&) = default;
};

```

In UERecord.cpp

add it to the function parameters and

add it into the function by writing this line at the bottom of the function:

this->VAR_NAME = var_name;



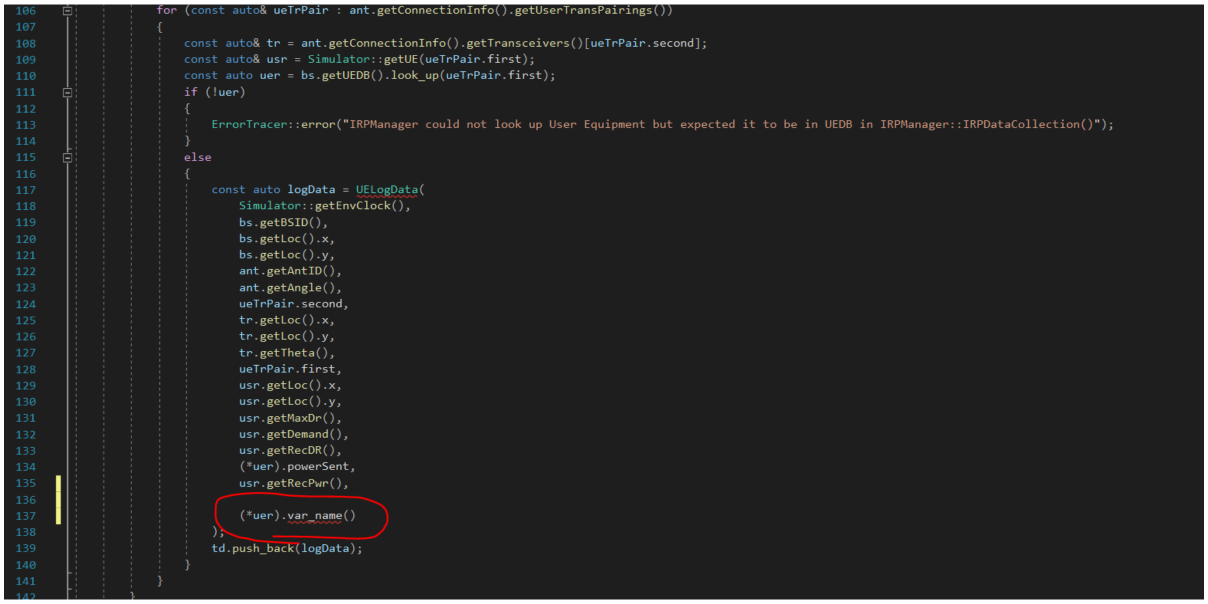
```cpp
#include "UERecord.h"

UERecord::UERecord(const size_t& uid, const Coord<float>& loc, const size_t& at, const size_t& ct, const float& cSNR, const uint32_t& d,
                   const uint32_t& bts, const float& ps, const datatype v_n)
{
    this->userID = uid;
    this->loc = Coord<float>(loc);
    this->antenna = at;
    this->currentTransceiver = ct;
    this->currentSNR = cSNR;
    this->demand = d;
    this->bitsSent = bts;
    this->powerSent = ps;

    this->var_name = v_n;
}
```

Note: Replace VAR_NAME with the actual name you want to use for the variable just like Step 2

Step 4: add this variable to IRPManager.cpp

This function collects the data from all the User Equipment Records every tick so you need to add your KPI to the list in order to get recorded.

the data type should be (*uer).var_name as the variable comes from UERecords



```
for (const auto& ueTrPair : ant.getConnectionInfo().getUserTransPairings())
{
    const auto& tr = ant.getConnectionInfo().getTransceivers()[ueTrPair.second];
    const auto& usr = Simulator::getUE(ueTrPair.first);
    const auto uer = bs.getUEDB().look_up(ueTrPair.first);
    if (!uer)
    {
        ErrorTracer::error("IRPManager could not look up User Equipment but expected it to be in UEDB in IRPManager::IRPDataCollection()");
    }
    else
    {
        const auto logData = UELogData(
            Simulator::getEnvClock(),
            bs.getBSID(),
            bs.getLoc().x,
            bs.getLoc().y,
            ant.getAntID(),
            ant.getAngle(),
            ueTrPair.second,
            tr.getLoc().x,
            tr.getLoc().y,
            tr.getTheta(),
            ueTrPair.first,
            usr.getLoc().x,
            usr.getLoc().y,
            usr.getMaxDr(),
            usr.getDemand(),
            usr.getRecDR(),
            (*uer).powerSent,
            usr.getRecPwr(),

            (*uer).var_name()
        );
        td.push_back(logData);
    }
}
```

Step 5: Adding this variable to FileIO.cpp

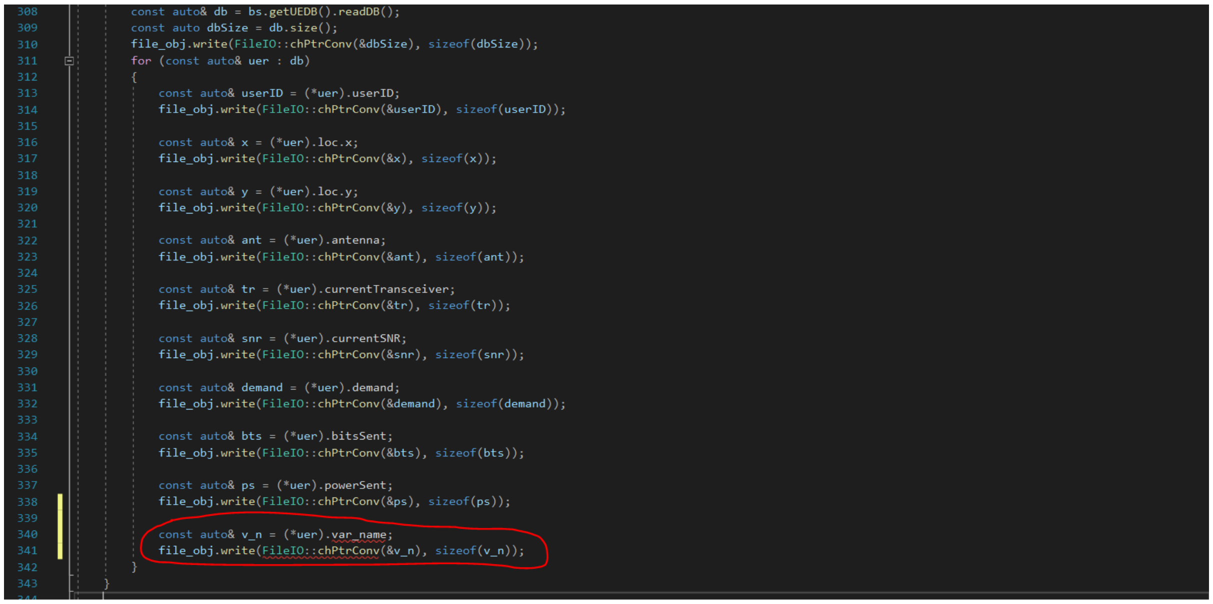
In FileIO::writeInitialSimulationState()

There is a for loop that goes though every User Equipment Record and writes their variables into a file object.

Add your variable into this for loop using the following 2 lines:

const auto& v_n = (*uer).var_name;

file_obj.write(FileIO::chPtrConv(&v_n),sizeof(v_n));



```
const auto& db = bs.getUEDB().readDB();
const auto dbSize = db.size();
file_obj.write(FileIO::chPtrConv(&dbSize), sizeof(dbSize));
for (const auto& uer : db)
{
    const auto& userID = (*uer).userID;
    file_obj.write(FileIO::chPtrConv(&userID), sizeof(userID));

    const auto& x = (*uer).loc.x;
    file_obj.write(FileIO::chPtrConv(&x), sizeof(x));

    const auto& y = (*uer).loc.y;
    file_obj.write(FileIO::chPtrConv(&y), sizeof(y));

    const auto& ant = (*uer).antenna;
    file_obj.write(FileIO::chPtrConv(&ant), sizeof(ant));

    const auto& tr = (*uer).currentTransceiver;
    file_obj.write(FileIO::chPtrConv(&tr), sizeof(tr));

    const auto& snr = (*uer).currentSNR;
    file_obj.write(FileIO::chPtrConv(&snr), sizeof(snr));

    const auto& demand = (*uer).demand;
    file_obj.write(FileIO::chPtrConv(&demand), sizeof(demand));

    const auto& bts = (*uer).bitsSent;
    file_obj.write(FileIO::chPtrConv(&bts), sizeof(bts));

    const auto& ps = (*uer).powerSent;
    file_obj.write(FileIO::chPtrConv(&ps), sizeof(ps));

    const auto& v_n = (*uer).var_name;
    file_obj.write(FileIO::chPtrConv(&v_n), sizeof(v_n));
}
}
```

Note: Replace v_n and var_name with the actual name you want to use for your variable

In FileIO::readSaveFileIntoSim()

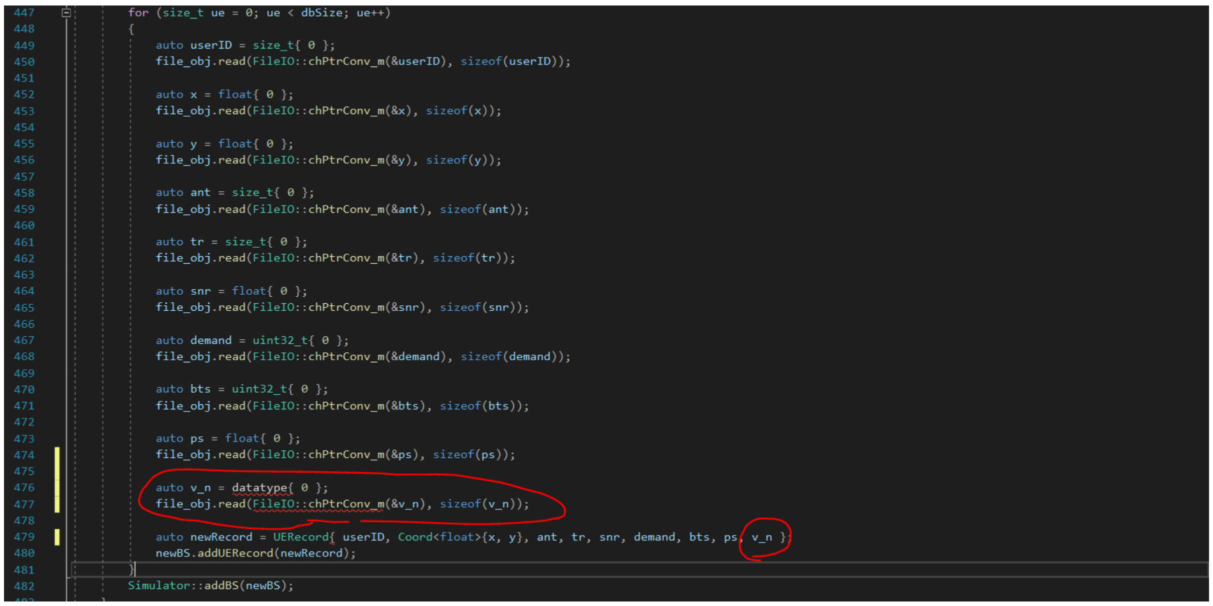
There is a for loop that reads the values from each User Equipment connected to a base station

Add your variable to this for loop using the following 2 lines:

auto v_n = datatype{ 0 };

file_obj.read(FileIO::chPtrConv_m(&v_n),sizeof(v_n));

Also add your varible to the auto newRecord = UERecord{ } function call

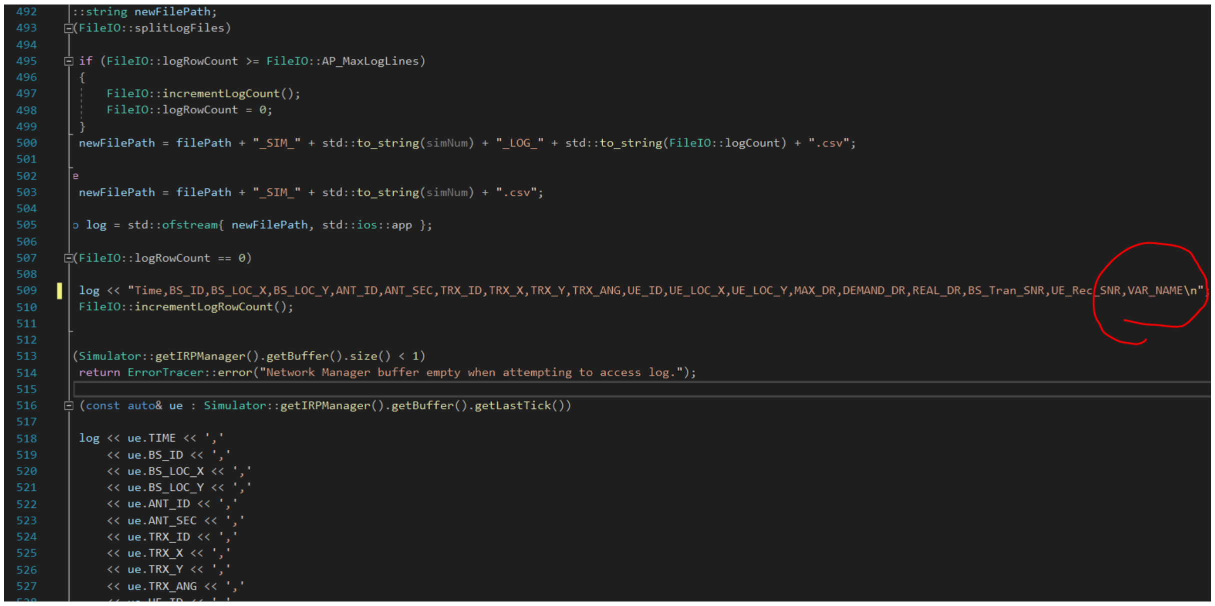


```
        for (size_t ue = 0; ue < dbSize; ue++)
        {
            auto userID = size_t{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&userID), sizeof(userID));

            auto x = float{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&x), sizeof(x));

            auto y = float{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&y), sizeof(y));

            auto ant = size_t{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&ant), sizeof(ant));

            auto tr = size_t{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&tr), sizeof(tr));

            auto snr = float{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&snr), sizeof(snr));

            auto demand = uint32_t{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&demand), sizeof(demand));

            auto bts = uint32_t{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&bts), sizeof(bts));

            auto ps = float{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&ps), sizeof(ps));

            auto v_n = datatype{ 0 };
            file_obj.read(FileIO::chPtrConv_m(&v_n), sizeof(v_n));

            auto newRecord = UERecord{ userID, Coord<float>{x, y}, ant, tr, snr, demand, bts, ps, v_n };
            newBS.addUERecord(newRecord);
        }
        Simulator::addBS(newBS);
```

Note: Replace v_n with the actual name you want to use for your variable and replace datatype with whatever datatype your variable is: most likely float or uint32_t

In FileIO::appendLog

Add your variable to the log here separated by a comma:

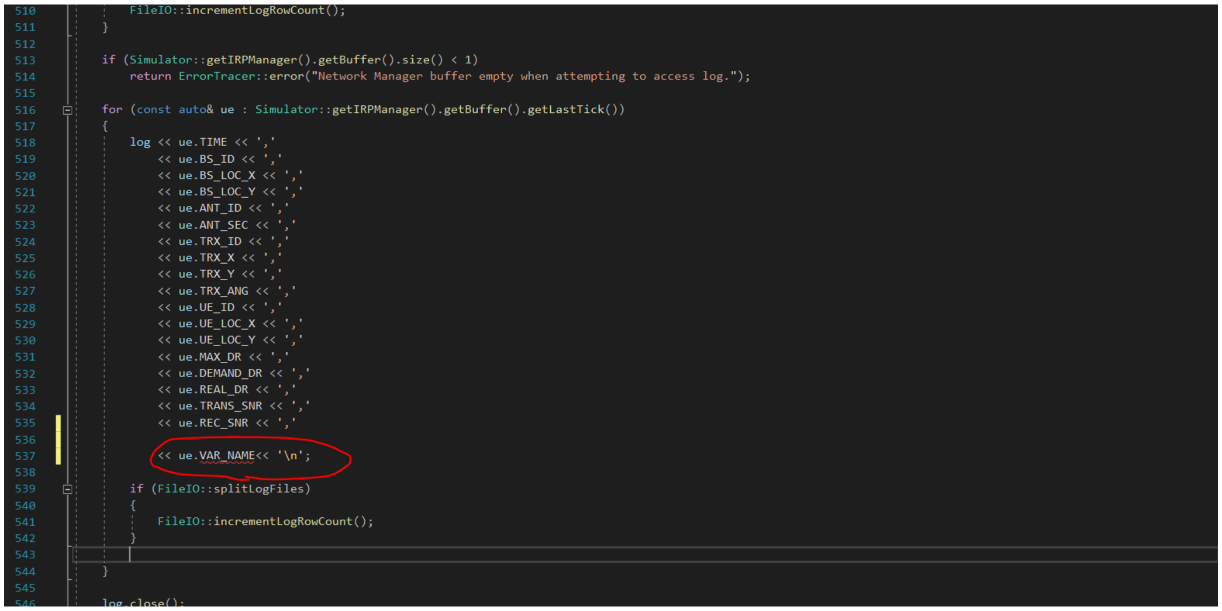


```
::string newFilePath;
(FileIO::splitLogFiles)

 if (FileIO::logRowCount >= FileIO::AP_MaxLogLines)
 {
     FileIO::incrementLogCount();
     FileIO::logRowCount = 0;
 }
 newFilePath = filePath + "_SIM_" + std::to_string(simNum) + "_LOG_" + std::to_string(FileIO::logCount) + ".csv";

e
 newFilePath = filePath + "_SIM_" + std::to_string(simNum) + ".csv";

o log = std::ofstream{ newFilePath, std::ios::app };

(FileIO::logRowCount == 0)

 log << "Time,BS_ID,BS_LOC_X,BS_LOC_Y,ANT_ID,ANT_SEC,TRX_ID,TRX_X,TRX_Y,TRX_ANG,UE_ID,UE_LOC_X,UE_LOC_Y,MAX_DR,DEMAND_DR,REAL_DR,BS_Tran_SNR,UE_Rec_SNR,VAR_NAME\n"
 FileIO::incrementLogRowCount();


(Simulator::getIRPManager().getBuffer().size() < 1)
 return ErrorTracer::error("Network Manager buffer empty when attempting to access log.");

 (const auto& ue : Simulator::getIRPManager().getBuffer().getLastTick())

 log << ue.TIME << ','
     << ue.BS_ID << ','
     << ue.BS_LOC_X << ','
     << ue.BS_LOC_Y << ','
     << ue.ANT_ID << ','
     << ue.ANT_SEC << ','
     << ue.TRX_ID << ','
     << ue.TRX_X << ','
     << ue.TRX_Y << ','
     << ue.TRX_ANG << ','
```

Note: Whatever you name the variable here will be how it appears on the csv file

Also add your variable to the for loop that adds each new line of data by adding the following line:

<< ue.VAR_NAME << ‘\n’;



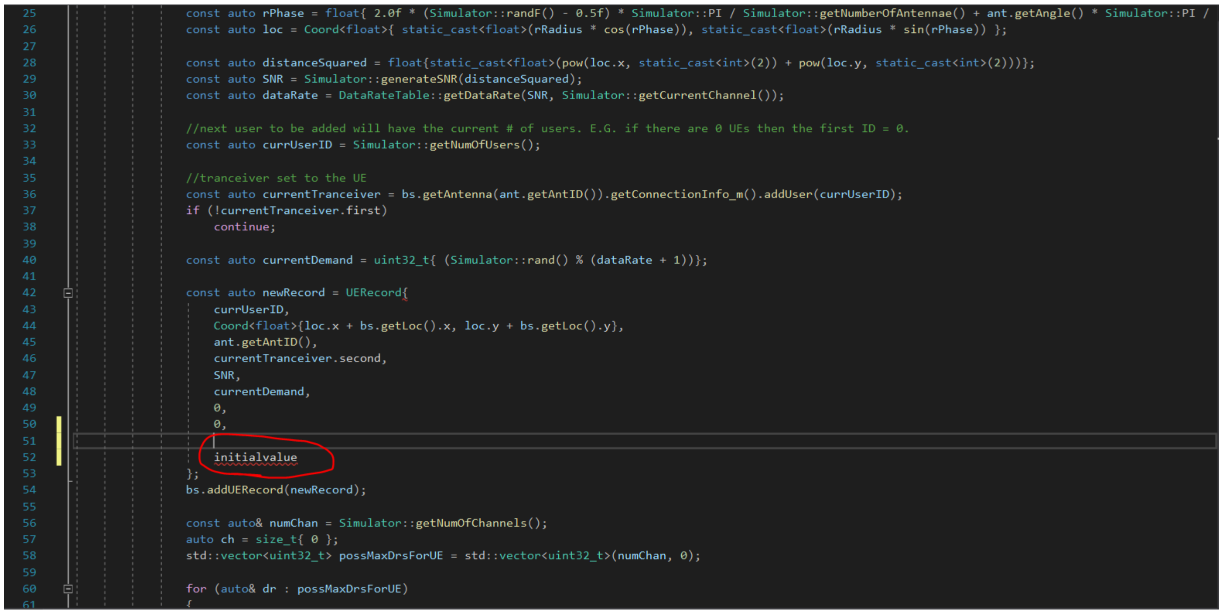
```
        FileIO::incrementLogRowCount();
    }

    if (Simulator::getIRPManager().getBuffer().size() < 1)
        return ErrorTracer::error("Network Manager buffer empty when attempting to access log.");

    for (const auto& ue : Simulator::getIRPManager().getBuffer().getLastTick())
    {
        log << ue.TIME << ','
            << ue.BS_ID << ','
            << ue.BS_LOC_X << ','
            << ue.BS_LOC_Y << ','
            << ue.ANT_ID << ','
            << ue.ANT_SEC << ','
            << ue.TRX_ID << ','
            << ue.TRX_X << ','
            << ue.TRX_Y << ','
            << ue.TRX_ANG << ','
            << ue.UE_ID << ','
            << ue.UE_LOC_X << ','
            << ue.UE_LOC_Y << ','
            << ue.MAX_DR << ','
            << ue.DEMAND_DR << ','
            << ue.REAL_DR << ','
            << ue.TRANS_SNR << ','
            << ue.REC_SNR << ','

            << ue.VAR_NAME<< '\n';

        if (FileIO::splitLogFiles)
        {
            FileIO::incrementLogRowCount();
        }

    }

    log.close();
```

Note: You cannot copy-paste this line as ‘ are treated differently by Visual Studio

Step 6: add the variable to EnvironmentalInitialization.cpp

In EnvironmentInitialization::setDefaultUsers()

This function initializes the environment so you need to add additional 0s, or a specific initial value if you need one, to this newRecord for each of the variables you are adding

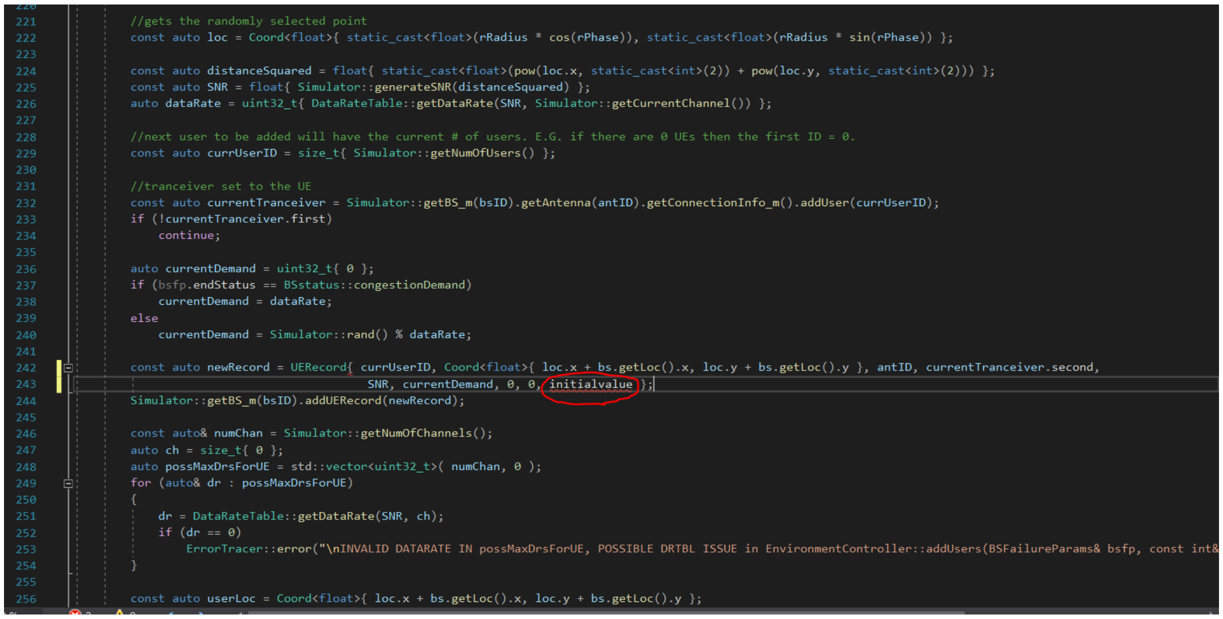


```cpp
const auto rPhase = float{ 2.0f * (Simulator::randF() - 0.5f) * Simulator::PI / Simulator::getNumberOfAntennae() + ant.getAngle() * Simulator::PI /
const auto loc = Coord<float>{ static_cast<float>(rRadius * cos(rPhase)), static_cast<float>(rRadius * sin(rPhase)) };

const auto distanceSquared = float{static_cast<float>(pow(loc.x, static_cast<int>(2)) + pow(loc.y, static_cast<int>(2)))};
const auto SNR = Simulator::generateSNR(distanceSquared);
const auto dataRate = DataRateTable::getDataRate(SNR, Simulator::getCurrentChannel());

//next user to be added will have the current # of users. E.G. if there are 0 UEs then the first ID = 0.
const auto currUserID = Simulator::getNumOfUsers();

//tranceiver set to the UE
const auto currentTranceiver = bs.getAntenna(ant.getAntID()).getConnectionInfo_m().addUser(currUserID);
if (!currentTranceiver.first)
    continue;

const auto currentDemand = uint32_t{ (Simulator::rand() % (dataRate + 1))};

const auto newRecord = UERecord{
    currUserID,
    Coord<float>{loc.x + bs.getLoc().x, loc.y + bs.getLoc().y},
    ant.getAntID(),
    currentTranceiver.second,
    SNR,
    currentDemand,
    0,
    0,

    initialvalue
};
bs.addUERecord(newRecord);

const auto& numChan = Simulator::getNumOfChannels();
auto ch = size_t{ 0 };
std::vector<uint32_t> possMaxDrsForUE = std::vector<uint32_t>(numChan, 0);

for (auto& dr : possMaxDrsForUE)
{
```

Step 7: add the variable to EnvironmentController.cpp

In EnvironmentController::addUsers()

Similar to Environment Initialization this function creates a default UERecord so you will need to add additional 0s, or initial values, for each of variables you are adding



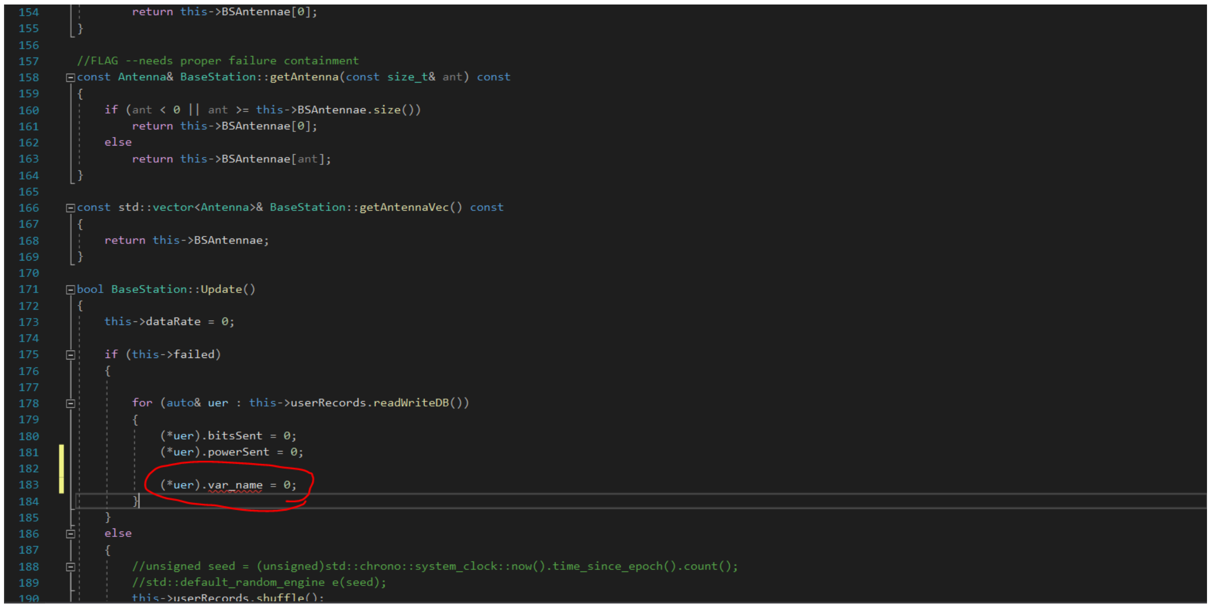
```cpp
//gets the randomly selected point
const auto loc = Coord<float>{ static_cast<float>(rRadius * cos(rPhase)), static_cast<float>(rRadius * sin(rPhase)) };

const auto distanceSquared = float{ static_cast<float>(pow(loc.x, static_cast<int>(2)) + pow(loc.y, static_cast<int>(2))) };
const auto SNR = float{ Simulator::generateSNR(distanceSquared) };
auto dataRate = uint32_t{ DataRateTable::getDataRate(SNR, Simulator::getCurrentChannel()) };

//next user to be added will have the current # of users. E.G. if there are 0 UEs then the first ID = 0.
const auto currUserID = size_t{ Simulator::getNumOfUsers() };

//tranceiver set to the UE
const auto currentTranceiver = Simulator::getBS_m(bsID).getAntenna(antID).getConnectionInfo_m().addUser(currUserID);
if (!currentTranceiver.first)
    continue;

auto currentDemand = uint32_t{ 0 };
if (bsfp.endStatus == BSstatus::congestionDemand)
    currentDemand = dataRate;
else
    currentDemand = Simulator::rand() % dataRate;

const auto newRecord = UERecord{ currUserID, Coord<float>{ loc.x + bs.getLoc().x, loc.y + bs.getLoc().y }, antID, currentTranceiver.second,
                                  SNR, currentDemand, 0, 0, initialvalue };
Simulator::getBS_m(bsID).addUERecord(newRecord);

const auto& numChan = Simulator::getNumOfChannels();
auto ch = size_t{ 0 };
auto possMaxDrsForUE = std::vector<uint32_t>( numChan, 0 );
for (auto& dr : possMaxDrsForUE)
{
    dr = DataRateTable::getDataRate(SNR, ch);
    if (dr == 0)
        ErrorTracer::error("\nINVALID DATARATE IN possMaxDrsForUE, POSSIBLE DRTBL ISSUE in EnvironmentController::addUsers(BSFailureParams& bsfp, const int&
}

const auto userLoc = Coord<float>{ loc.x + bs.getLoc().x, loc.y + bs.getLoc().y };
```

Step 8: add the variable to BaseStation.cpp

In BaseStation::Update()

Add a failure condition for each variable by adding this line to the first for loop

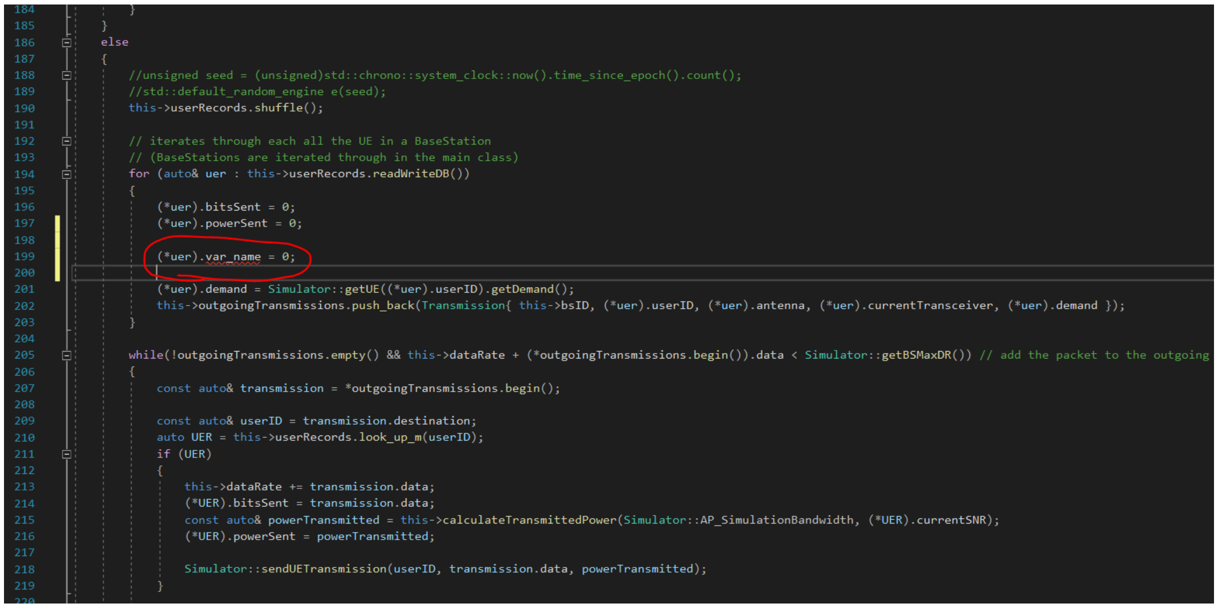
(*uer).var_name = 0;



```cpp
        return this->BSAntennae[0];
}

//FLAG --needs proper failure containment
const Antenna& BaseStation::getAntenna(const size_t& ant) const
{
    if (ant < 0 || ant >= this->BSAntennae.size())
        return this->BSAntennae[0];
    else
        return this->BSAntennae[ant];
}

const std::vector<Antenna>& BaseStation::getAntennaVec() const
{
    return this->BSAntennae;
}

bool BaseStation::Update()
{
    this->dataRate = 0;

    if (this->failed)
    {

        for (auto& uer : this->userRecords.readWriteDB())
        {
            (*uer).bitsSent = 0;
            (*uer).powerSent = 0;

            (*uer).var_name = 0;
        }
    }
    else
    {
        //unsigned seed = (unsigned)std::chrono::system_clock::now().time_since_epoch().count();
        //std::default_random_engine e(seed);
```

Then add the initial condition to the following for loop by adding the following line

(*uer).var_name = 0;

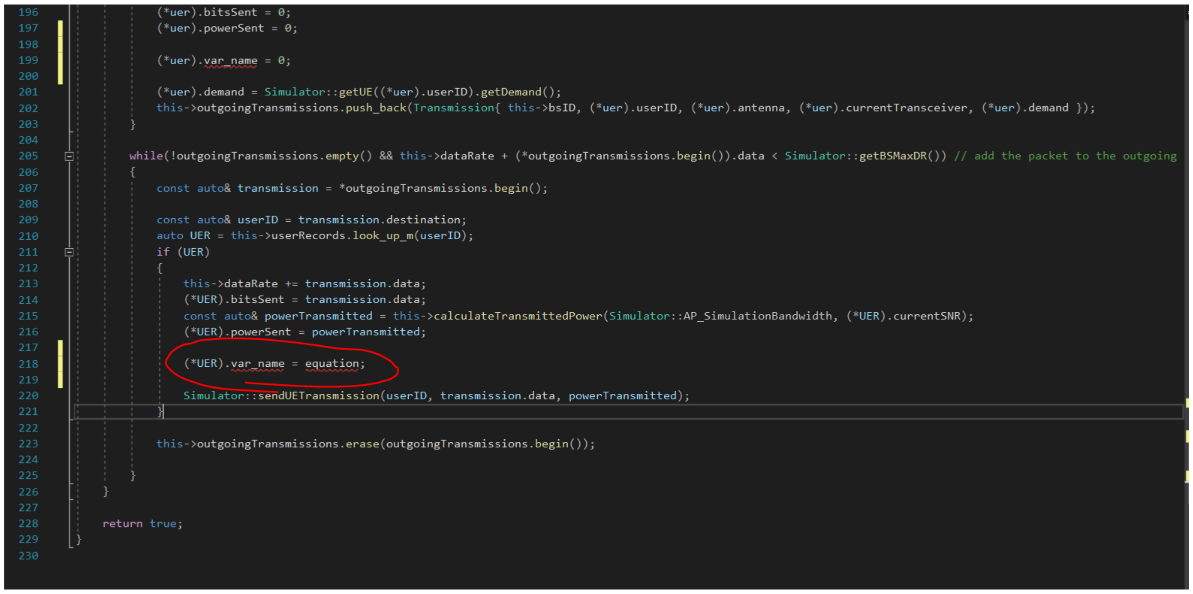


```cpp
        }
    }
    else
    {
        //unsigned seed = (unsigned)std::chrono::system_clock::now().time_since_epoch().count();
        //std::default_random_engine e(seed);
        this->userRecords.shuffle();

        // iterates through each all the UE in a BaseStation
        // (BaseStations are iterated through in the main class)
        for (auto& uer : this->userRecords.readWriteDB())
        {
            (*uer).bitsSent = 0;
            (*uer).powerSent = 0;

            (*uer).var_name = 0;

            (*uer).demand = Simulator::getUE((*uer).userID).getDemand();
            this->outgoingTransmissions.push_back(Transmission{ this->bsID, (*uer).userID, (*uer).antenna, (*uer).currentTransceiver, (*uer).demand });
        }

        while(!outgoingTransmissions.empty() && this->dataRate + (*outgoingTransmissions.begin()).data < Simulator::getBSMaxDR()) // add the packet to the outgoing
        {
            const auto& transmission = *outgoingTransmissions.begin();

            const auto& userID = transmission.destination;
            auto UER = this->userRecords.look_up_m(userID);
            if (UER)
            {
                this->dataRate += transmission.data;
                (*UER).bitsSent = transmission.data;
                const auto& powerTransmitted = this->calculateTransmittedPower(Simulator::AP_SimulationBandwidth, (*UER).currentSNR);
                (*UER).powerSent = powerTransmitted;

                Simulator::sendUETransmission(userID, transmission.data, powerTransmitted);
            }
```

Finally you need to add the code that calculates the actual value for the KPI

This should be done in the if statement at the end of BaseStation::Update by adding the following line:

```
(*UER).var_name = equation ;
```



```
            (*uer).bitsSent = 0;
            (*uer).powerSent = 0;

            (*uer).var_name = 0;

            (*uer).demand = Simulator::getUE((*uer).userID).getDemand();
            this->outgoingTransmissions.push_back(Transmission{ this->bsID, (*uer).userID, (*uer).antenna, (*uer).currentTransceiver, (*uer).demand });
        }

        while(!outgoingTransmissions.empty() && this->dataRate + (*outgoingTransmissions.begin()).data < Simulator::getBSMaxDR()) // add the packet to the outgoing
        {
            const auto& transmission = *outgoingTransmissions.begin();

            const auto& userID = transmission.destination;
            auto UER = this->userRecords.look_up_m(userID);
            if (UER)
            {
                this->dataRate += transmission.data;
                (*UER).bitsSent = transmission.data;
                const auto& powerTransmitted = this->calculateTransmittedPower(Simulator::AP_SimulationBandwidth, (*UER).currentSNR);
                (*UER).powerSent = powerTransmitted;

                (*UER).var_name = equation;

                Simulator::sendUETransmission(userID, transmission.data, powerTransmitted);
            }

            this->outgoingTransmissions.erase(outgoingTransmissions.begin());

        }
    }

    return true;
}
```

Now you are done

If you run the simulator now you should see a new column of values on the .csv file for your new KPI