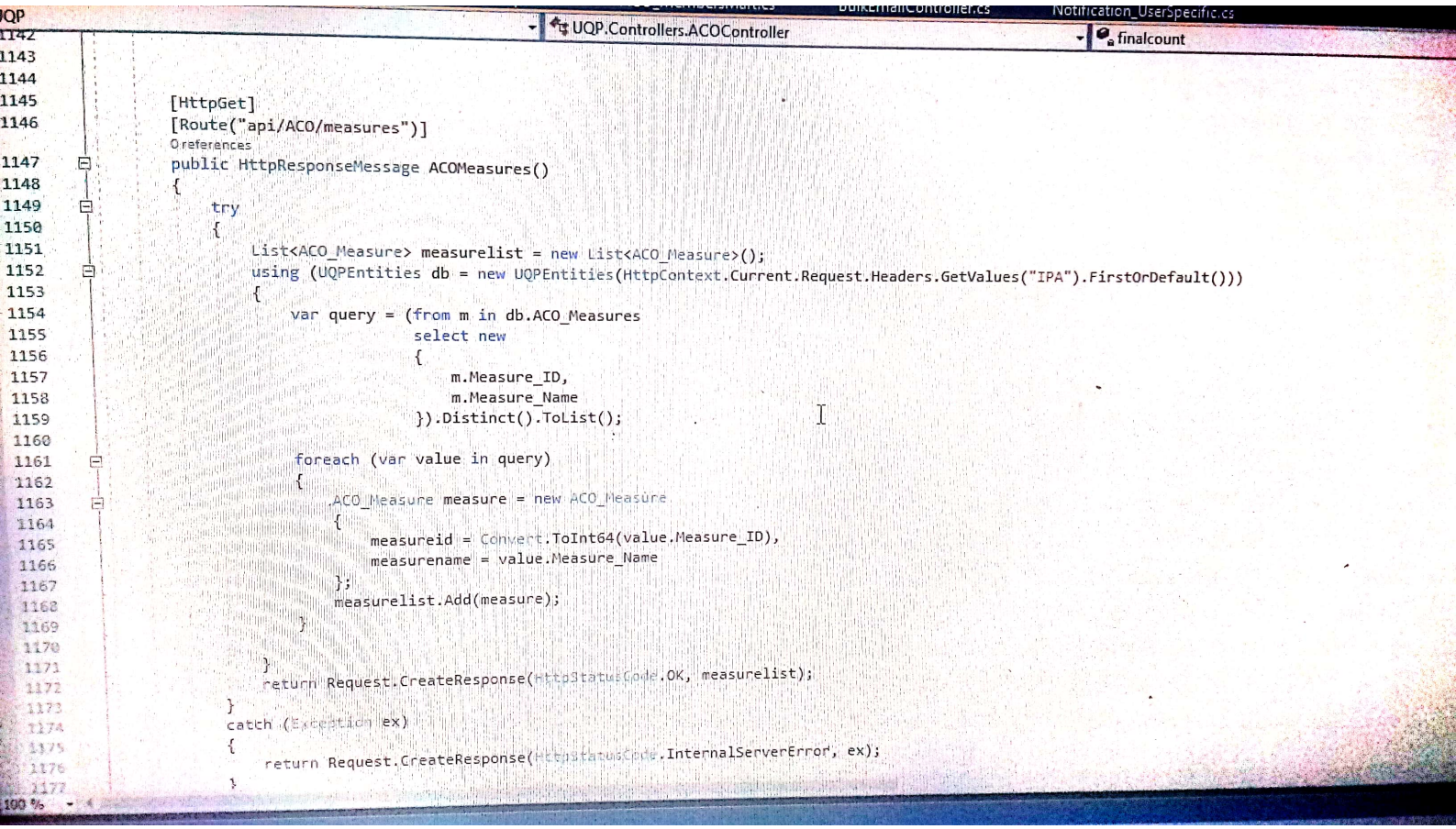
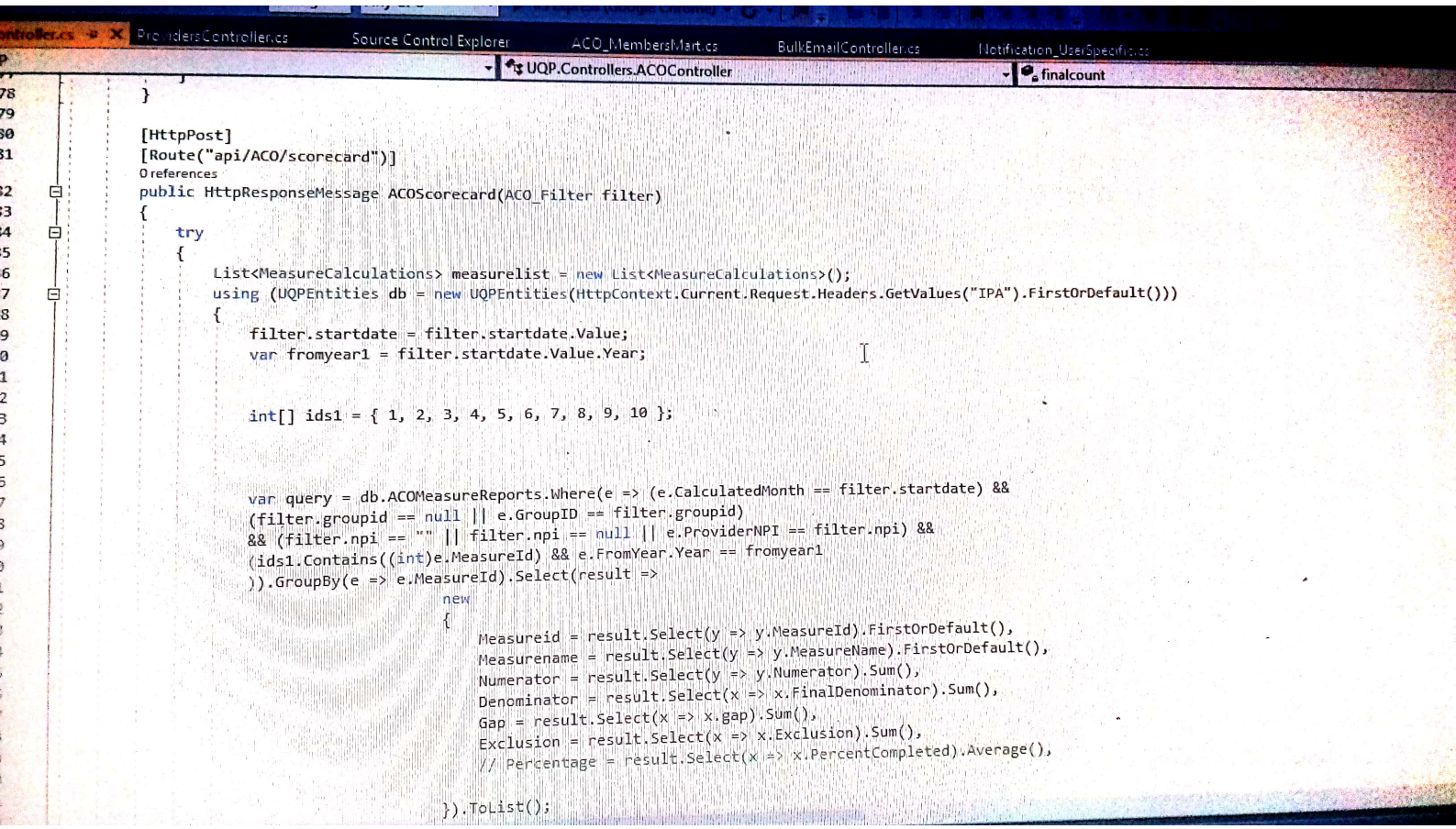
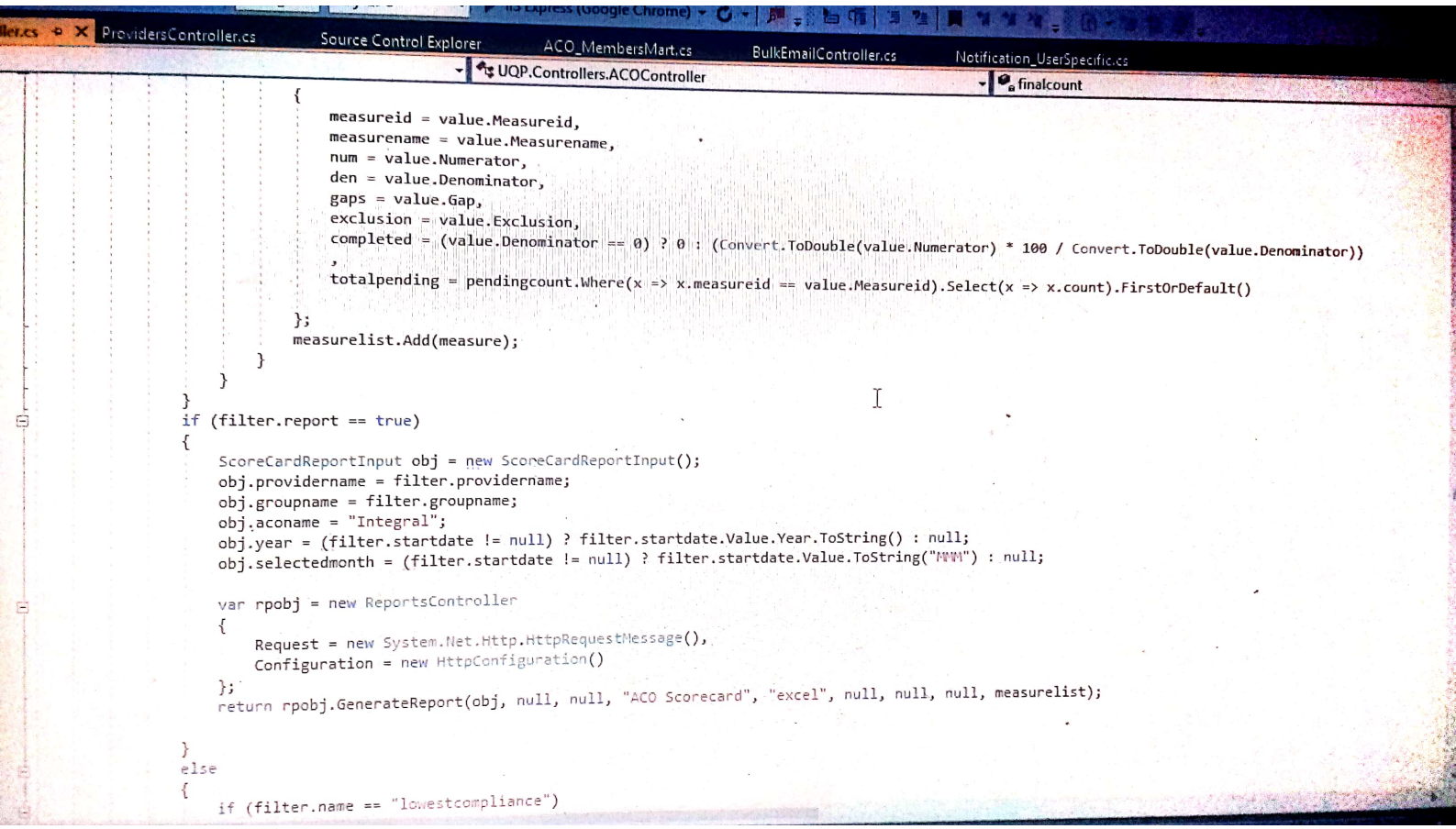
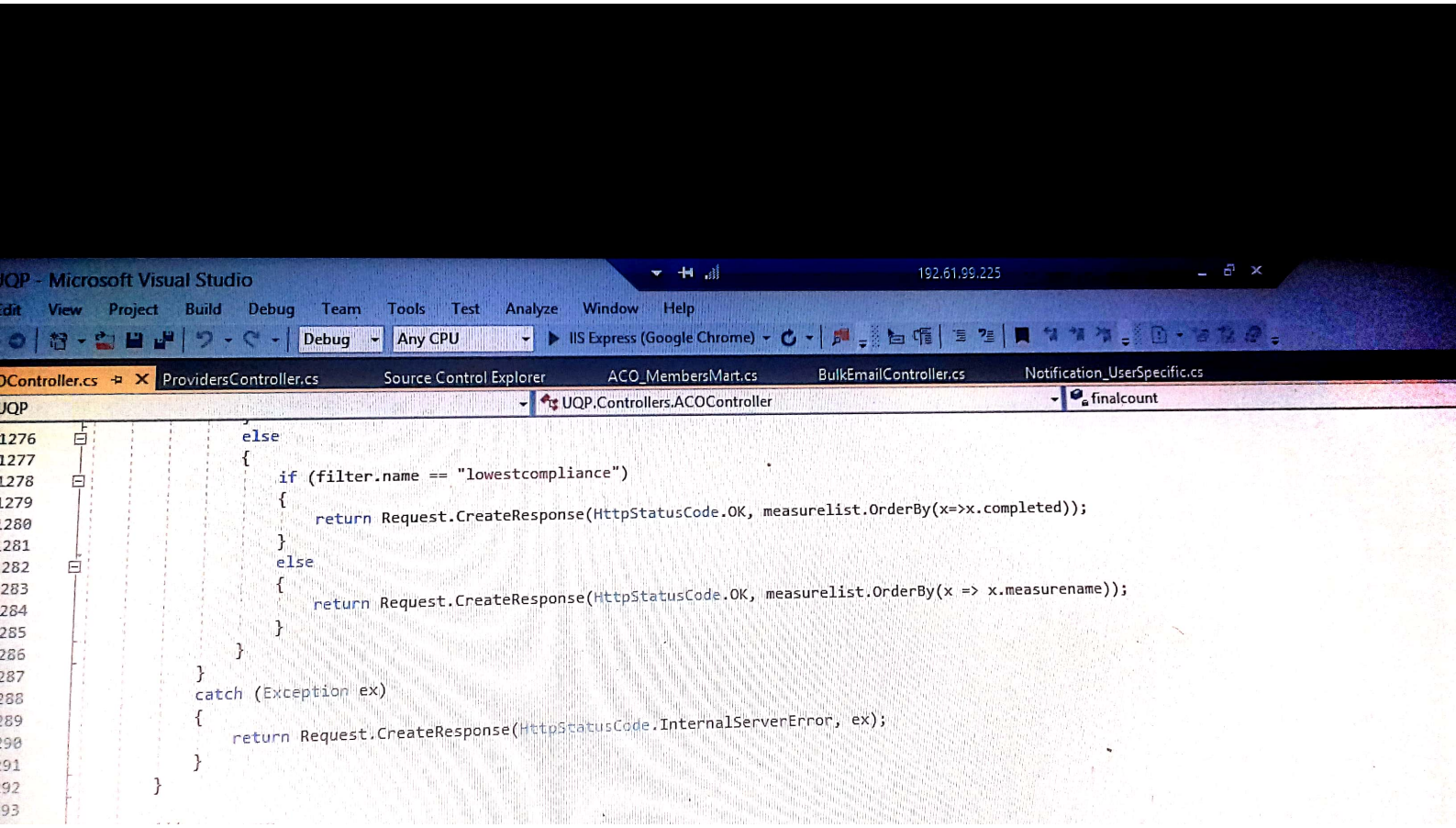
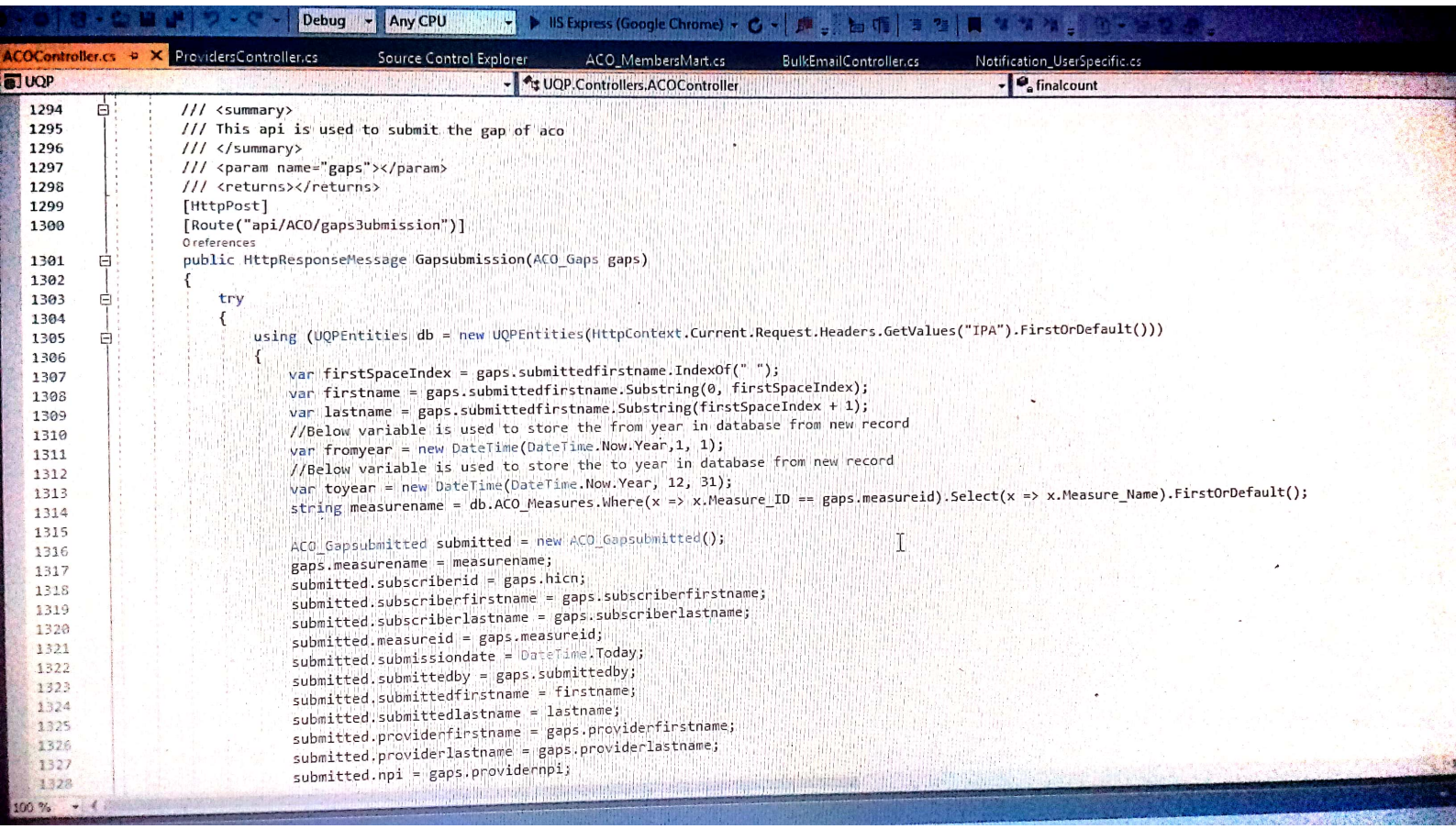


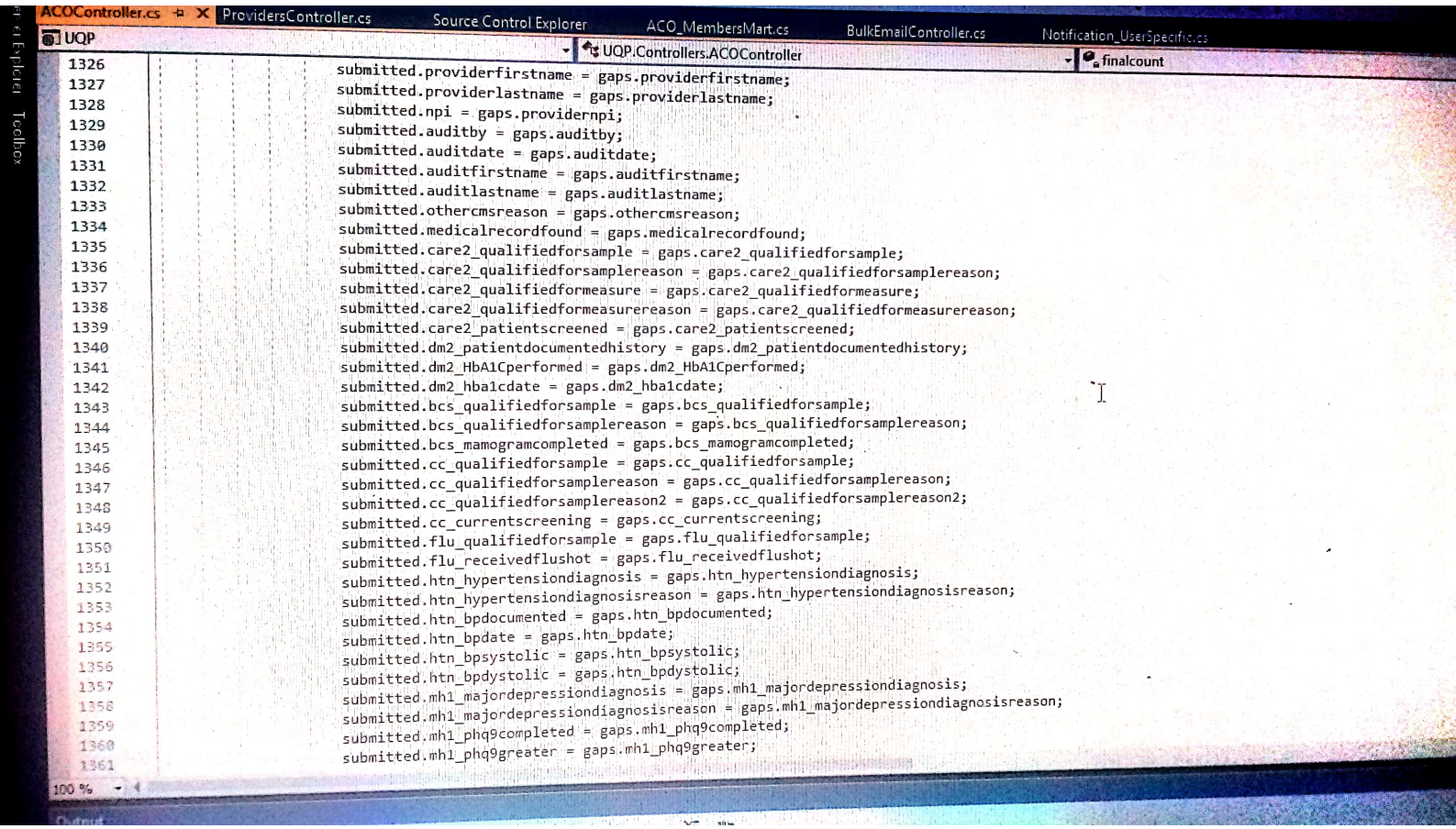


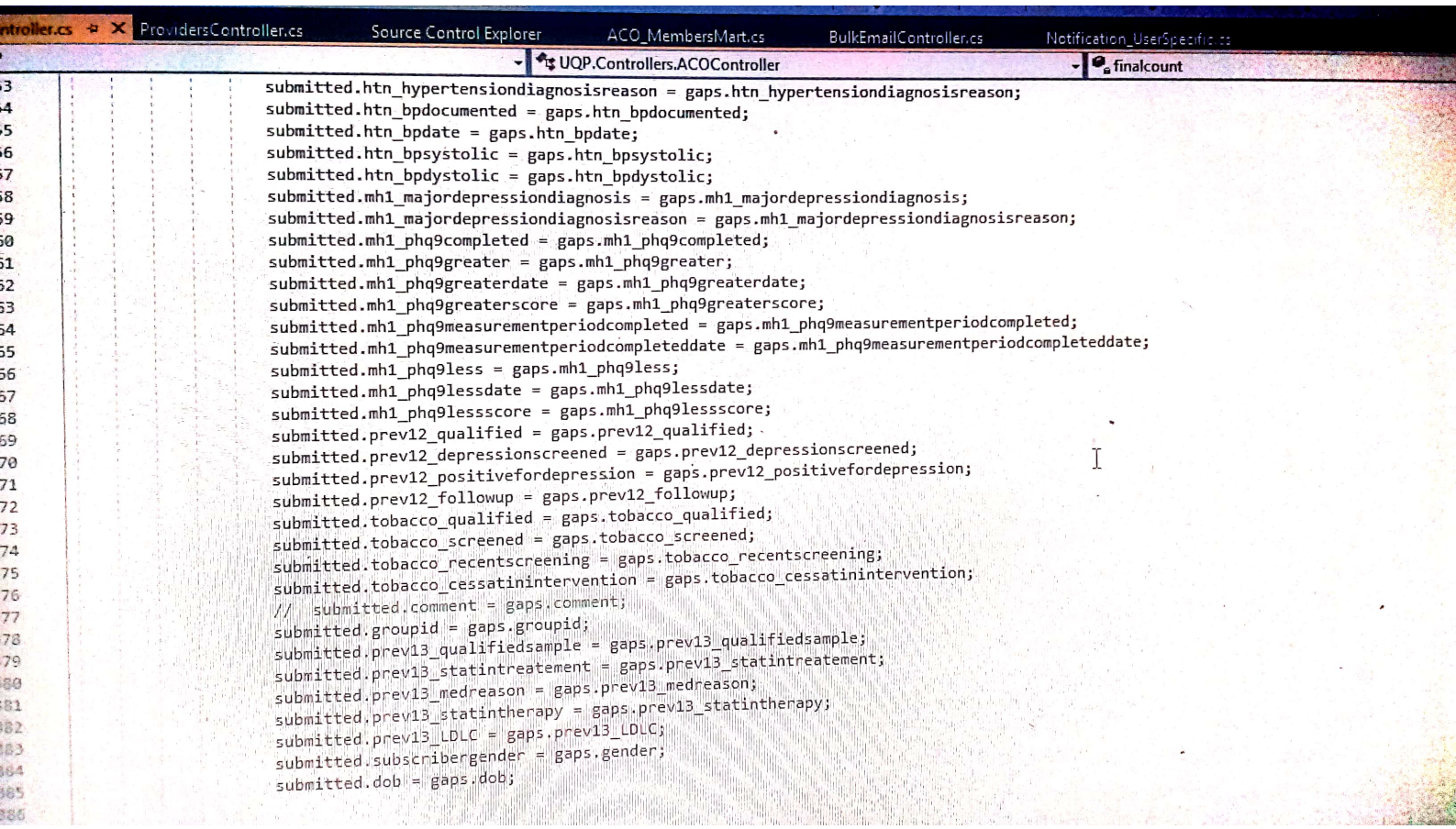
```
                                                                    finalcount
                                // Percentage = result.Select(x => x.PercentCompleted).Average(),
                            }).ToList();

        var pendingcount = (from e in db.ACO_Gapsubmitted
                            join
fd in db.ACO_Measures_Members
on new { sid = e.subscriberid, mid = (int?)e.measureid } equals new { sid = fd.BENE_HIC_NUM, mid = (int?)fd.MeasureId }
                            where (e.submissiondate<= filter.enddate) &&
        (filter.groupid == null || fd.Group_ID == filter.groupid)
        && (filter.npi == "" || filter.npi == null || fd.ProviderNPI == filter.npi) &&
        (ids1.Contains((int)fd.MeasureId) && fd.fromyear.Value.Year == fromyear1
        )
                            && fd.documentstatus == "pending"
                            group new { e, fd } by new
                            {
                                fd.MeasureId
                            } into gr
                            select new
                            {
                                measureid = gr.Select(x => x.fd.MeasureId).FirstOrDefault(),
                                count = gr.Select(y => y.fd.BENE_HIC_NUM).Distinct().Count()
                            }).ToList();



        if (query.Count() != 0)
        {

            foreach (var value in query)
            {
                MeasureCalculations measure = new MeasureCalculations
                {
                    measureid = value.Measureid,
                    measurename = value.Measurename,
```

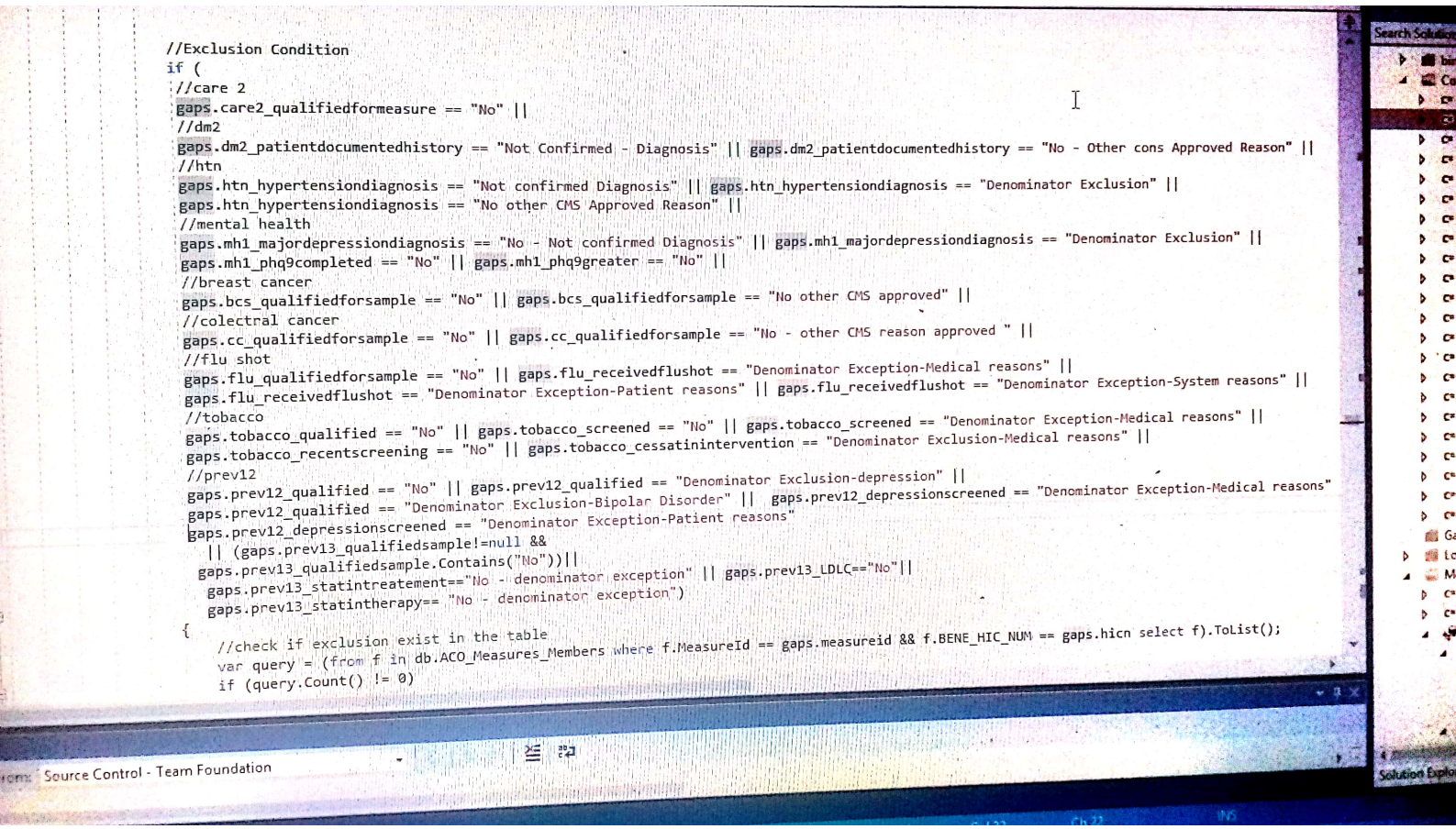








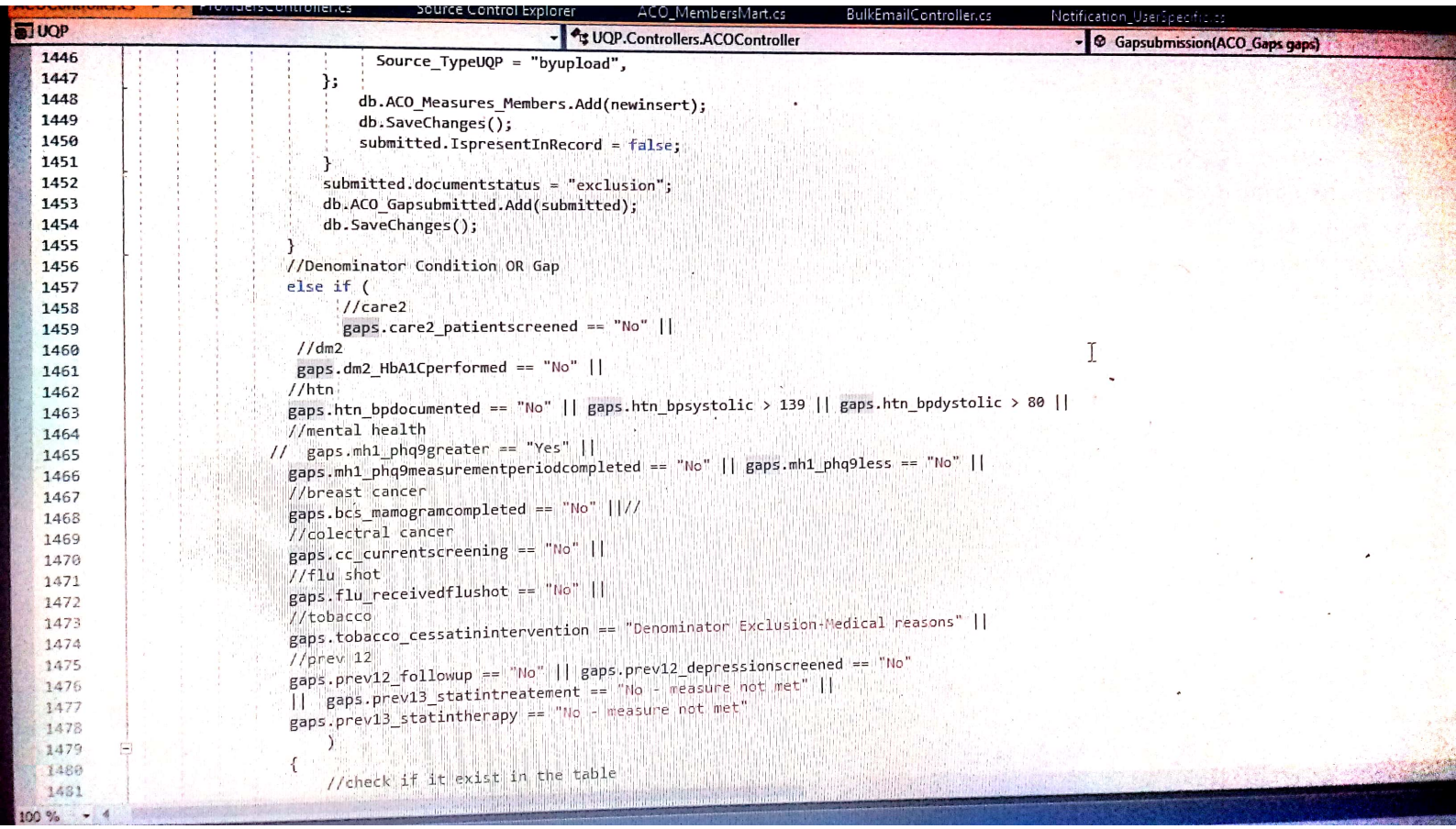


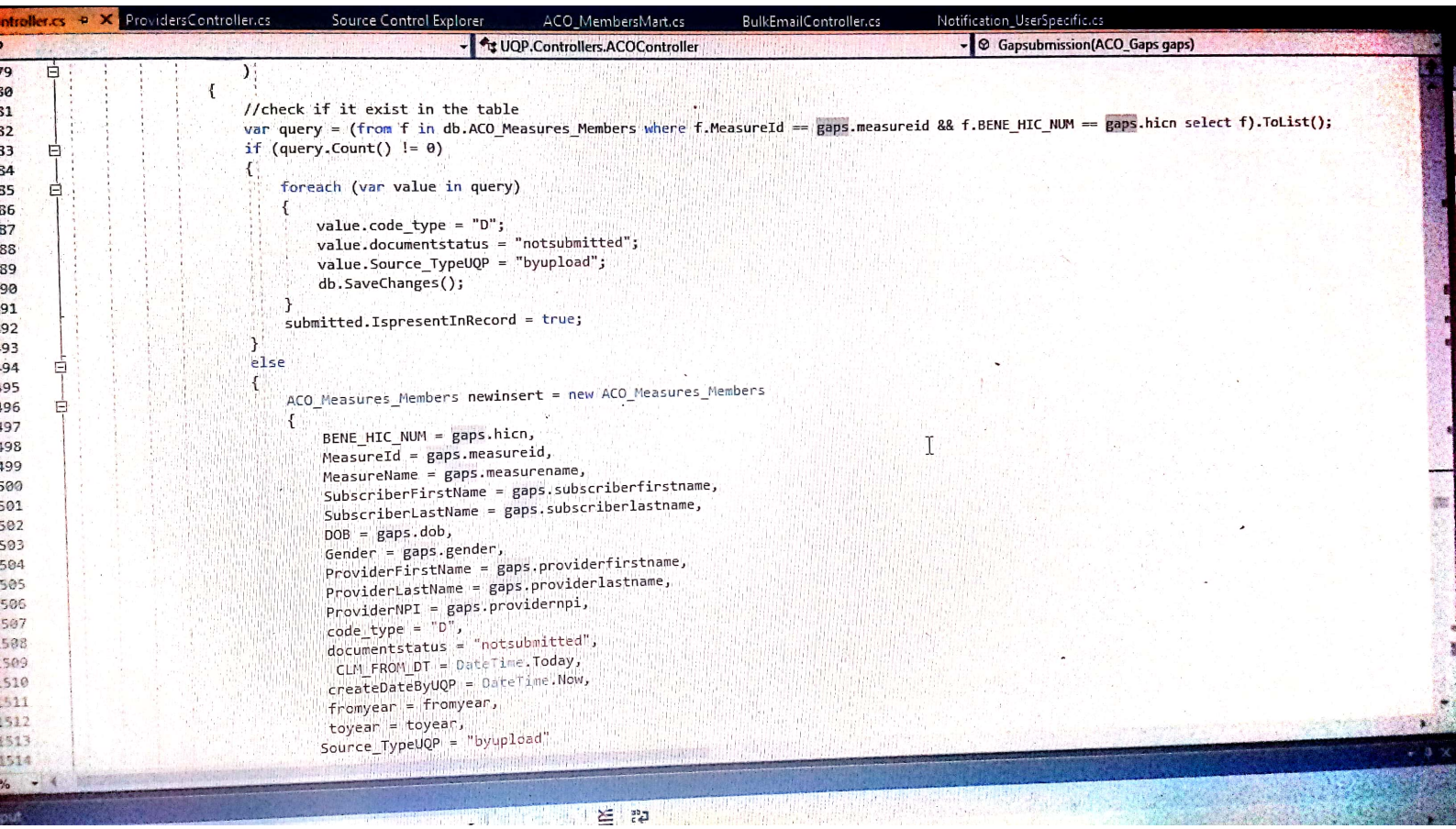


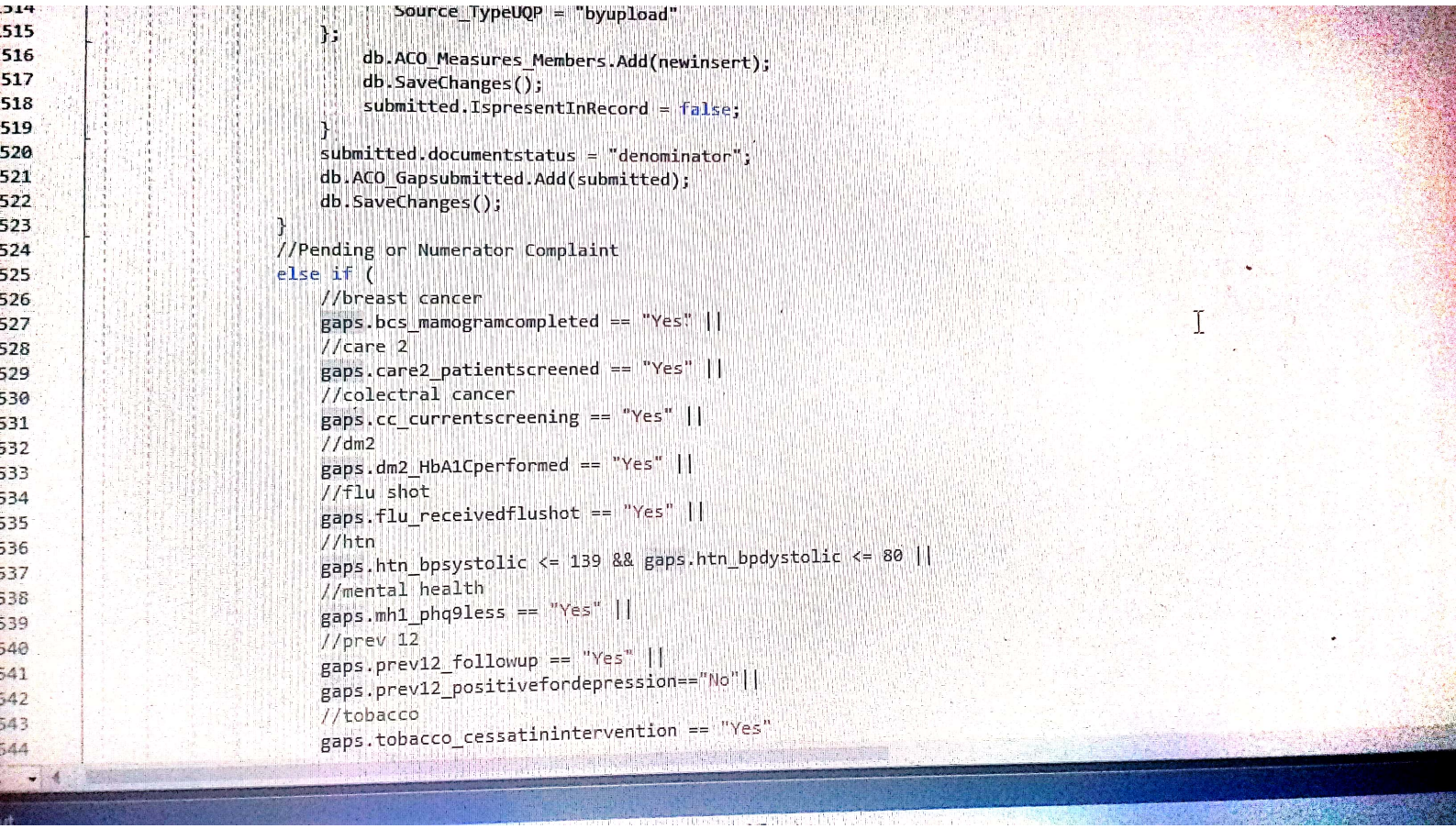
```
//Exclusion Condition
if (
 //care 2
 gaps.care2_qualifiedformeasure == "No" ||
 //dm2
 gaps.dm2_patientdocumentedhistory == "Not Confirmed - Diagnosis" || gaps.dm2_patientdocumentedhistory == "No - Other cons Approved Reason" ||
 //htn
 gaps.htn_hypertensiondiagnosis == "Not confirmed Diagnosis" || gaps.htn_hypertensiondiagnosis == "Denominator Exclusion" ||
 gaps.htn_hypertensiondiagnosis == "No other CMS Approved Reason" ||
 //mental health
 gaps.mh1_majordepressiondiagnosis == "No - Not confirmed Diagnosis" || gaps.mh1_majordepressiondiagnosis == "Denominator Exclusion" ||
 gaps.mh1_phq9completed == "No" || gaps.mh1_phq9greater == "No" ||
 //breast cancer
 gaps.bcs_qualifiedforsample == "No" || gaps.bcs_qualifiedforsample == "No other CMS approved" ||
 //colectral cancer
 gaps.cc_qualifiedforsample == "No" || gaps.cc_qualifiedforsample == "No - other CMS reason approved " ||
 //flu shot
 gaps.flu_qualifiedforsample == "No" || gaps.flu_receivedflushot == "Denominator Exception-Medical reasons" ||
 gaps.flu_receivedflushot == "Denominator Exception-Patient reasons" || gaps.flu_receivedflushot == "Denominator Exception-System reasons" ||
 //tobacco
 gaps.tobacco_qualified == "No" || gaps.tobacco_screened == "No" || gaps.tobacco_screened == "Denominator Exception-Medical reasons" ||
 gaps.tobacco_recentscreening == "No" || gaps.tobacco_cessatinintervention == "Denominator Exclusion-Medical reasons" ||
 //prev12
 gaps.prev12_qualified == "No" || gaps.prev12_qualified == "Denominator Exclusion-depression" ||
 gaps.prev12_qualified == "Denominator Exclusion-Bipolar Disorder" ||  gaps.prev12_depressionscreened == "Denominator Exception-Medical reasons"
 gaps.prev12_depressionscreened == "Denominator Exception-Patient reasons"
   || (gaps.prev13_qualifiedsample!=null &&
  gaps.prev13_qualifiedsample.Contains("No"))||
   gaps.prev13_statintreatement=="No - denominator exception" || gaps.prev13_LDLC=="No"||
   gaps.prev13_statintherapy== "No - denominator exception")
{
    //check if exclusion exist in the table
    var query = (from f in db.ACO_Measures_Members where f.MeasureId == gaps.measureid && f.BENE_HIC_NUM == gaps.hicn select f).ToList();
    if (query.Count() != 0)
```

Source Control - Team Foundation

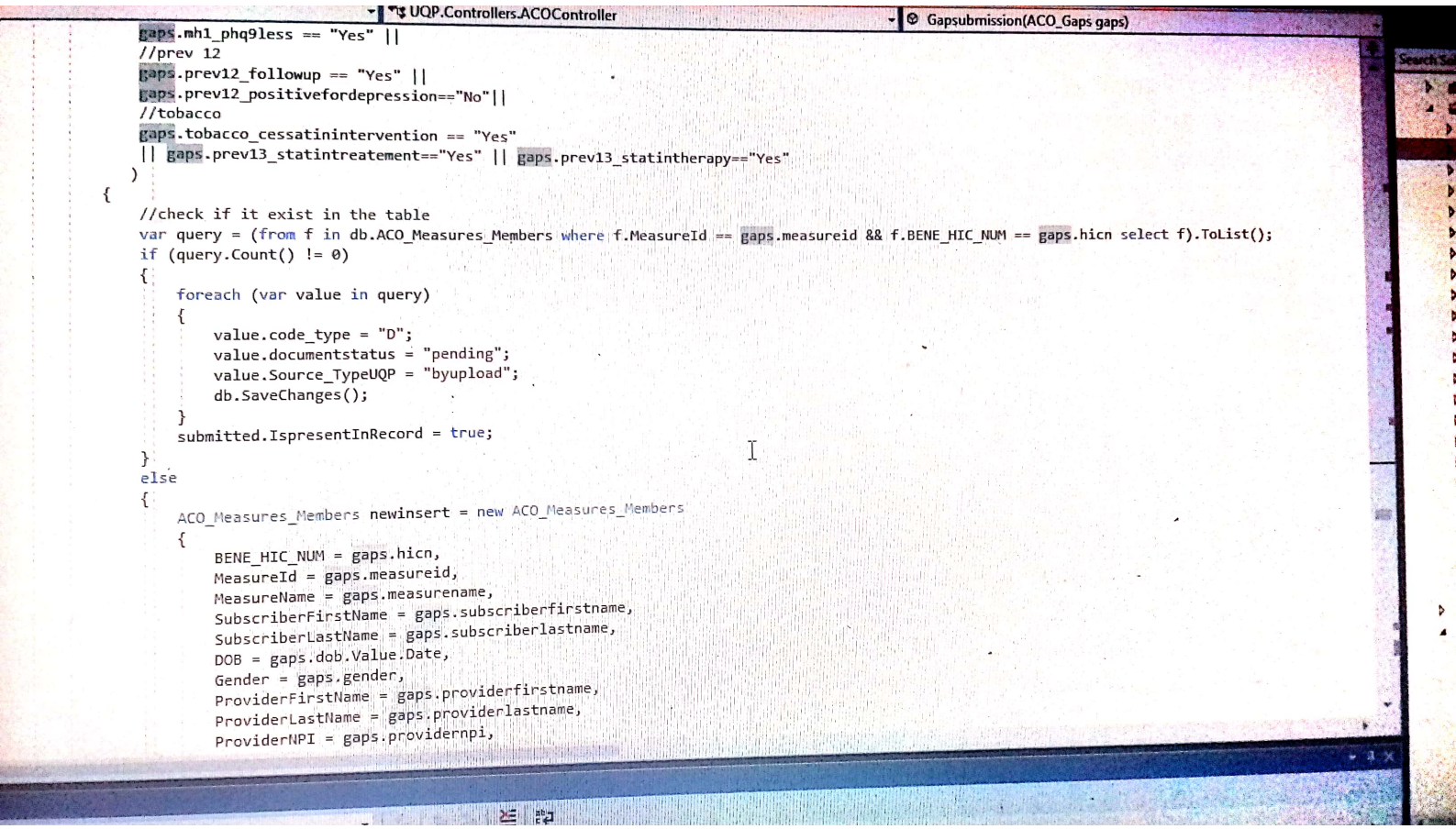
```
1413            //check if exclusion exist in the table
1414            var query = (from f in db.ACO_Measures_Members where f.MeasureId == gaps.measureid && f.BENE_HIC_NUM == gaps.hicn select f).ToList();
1415            if (query.Count() != 0)
1416            {
1417                foreach (var value in query)
1418                {
1419                    value.code_type = "E";
1420                    value.documentstatus = "exclusion";
1421                    value.Source_TypeUQP = "byupload";
1422                    db.SaveChanges();
1423                }
1424                submitted.IspresentInRecord = true;
1425            }
1426            else
1427            {
1428                ACO_Measures_Members newinsert = new ACO_Measures_Members
1429                {
1430                    BENE_HIC_NUM = gaps.hicn,
1431                    MeasureId = gaps.measureid,
1432                    MeasureName = gaps.measurename,
1433                    SubscriberFirstName = gaps.subscriberfirstname,
1434                    SubscriberLastName = gaps.subscriberlastname,
1435                    DOB = gaps.dob,
1436                    Gender = gaps.gender,
1437                    ProviderFirstName = gaps.providerfirstname,
1438                    ProviderLastName = gaps.providerlastname,
1439                    ProviderNPI = gaps.providernpi,
1440                    code_type = "E",
1441                    documentstatus = "exclusion",
1442                    CLM_FROM_DT = DateTime.Today,
1443                    createDateByUQP = DateTime.Now,
1444                    fromyear = fromyear,
1445                    toyear=toyear,
1446                    Source_TypeUQP = "byupload",
1447                };
1448                db.ACO_Measures_Members.Add(newinsert);
```



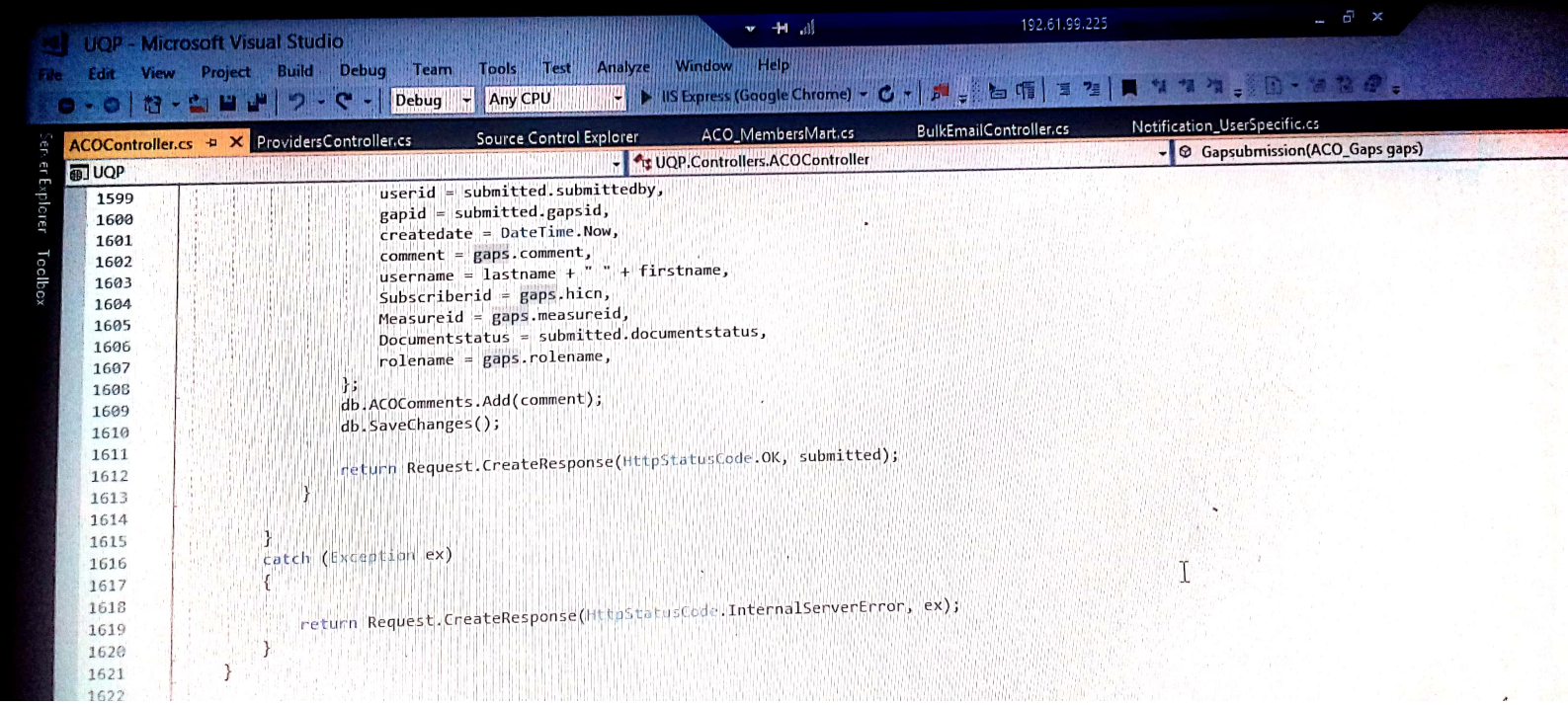


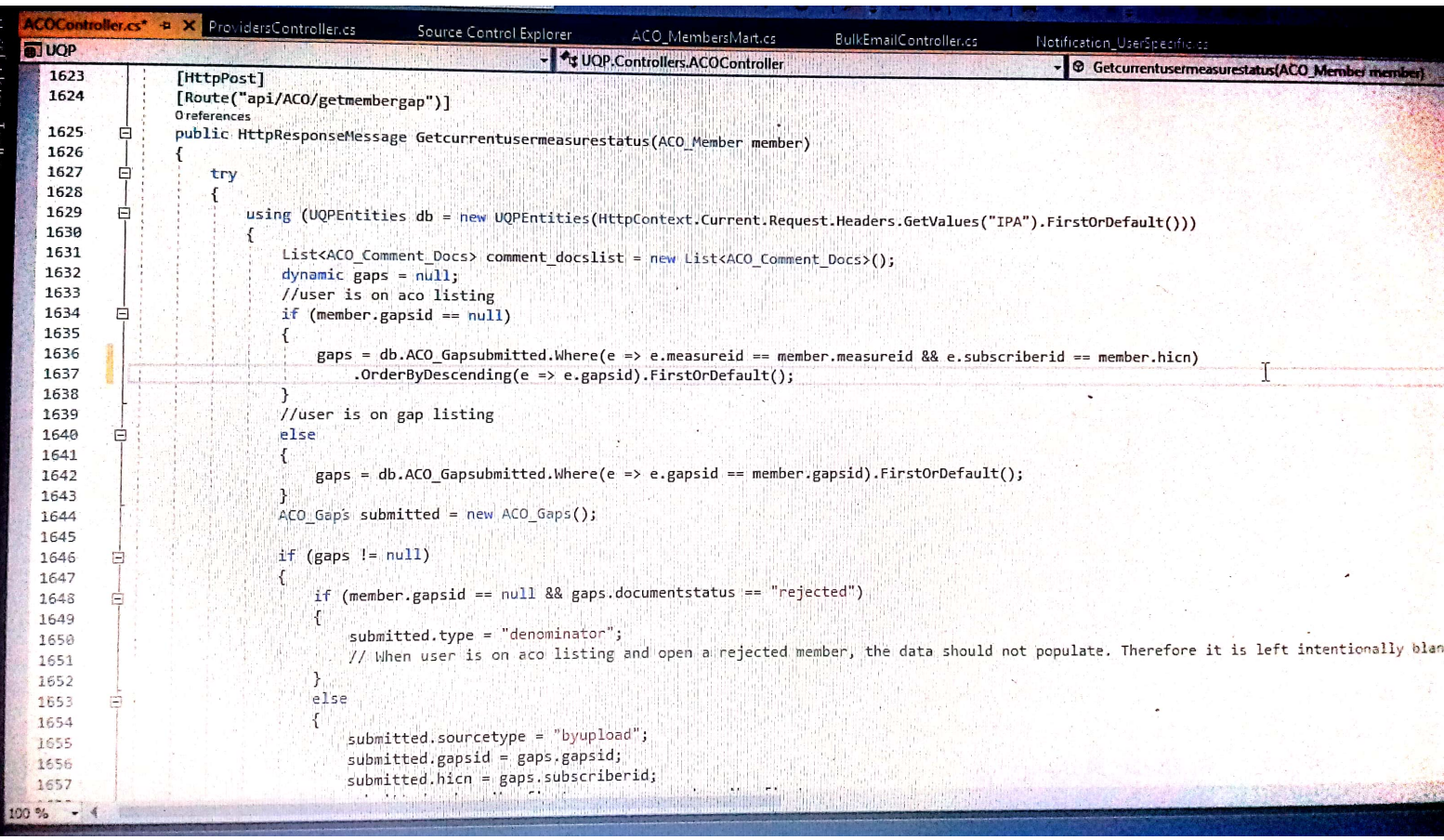


```
                Source_TypeUQP = "byupload"
            };
                db.ACO_Measures_Members.Add(newinsert);
                db.SaveChanges();
                submitted.IspresentInRecord = false;
            }
            submitted.documentstatus = "denominator";
            db.ACO_Gapsubmitted.Add(submitted);
            db.SaveChanges();
        }
        //Pending or Numerator Complaint
        else if (
            //breast cancer
            gaps.bcs_mamogramcompleted == "Yes" ||
            //care 2
            gaps.care2_patientscreened == "Yes" ||
            //colectral cancer
            gaps.cc_currentscreening == "Yes" ||
            //dm2
            gaps.dm2_HbA1Cperformed == "Yes" ||
            //flu shot
            gaps.flu_receivedflushot == "Yes" ||
            //htn
            gaps.htn_bpsystolic <= 139 && gaps.htn_bpdystolic <= 80 ||
            //mental health
            gaps.mh1_phq9less == "Yes" ||
            //prev 12
            gaps.prev12_followup == "Yes" ||
            gaps.prev12_positivefordepression=="No"||
            //tobacco
            gaps.tobacco_cessatinintervention == "Yes"
```



```
                ProviderFirstName = gaps.providerfirstname,
                ProviderLastName = gaps.providerlastname,
                ProviderNPI = gaps.providernpi,
                code_type = "D",
                documentstatus = "pending",
                CLM_FROM_DT = DateTime.Today,
                createDateByUQP = DateTime.Now,
                fromyear = fromyear,
                toyear = toyear,
                Source_TypeUQP = "byupload",
            };
            db.ACO_Measures_Members.Add(newinsert);
            db.SaveChanges();
            submitted.IspresentInRecord = false;
        }
        submitted.documentstatus = "pending";
        db.ACO_Gapsubmitted.Add(submitted);
        db.SaveChanges();
        submitted.gapsid = submitted.gapsid;


    }

    //It will maintain the history and it will also store the comment will run in every condition
    ACOComment comment = new ACOComment
    {
        userid = submitted.submittedby,
        gapid = submitted.gapsid,
        createdate = DateTime.Now,
        comment = gaps.comment,
        username = lastname + " " + firstname,
        Subscriberid = gaps.hicn,
        Measureid = gaps.measureid,
        Documentstatus = submitted.documentstatus,
        rolename = gaps.rolename,
```

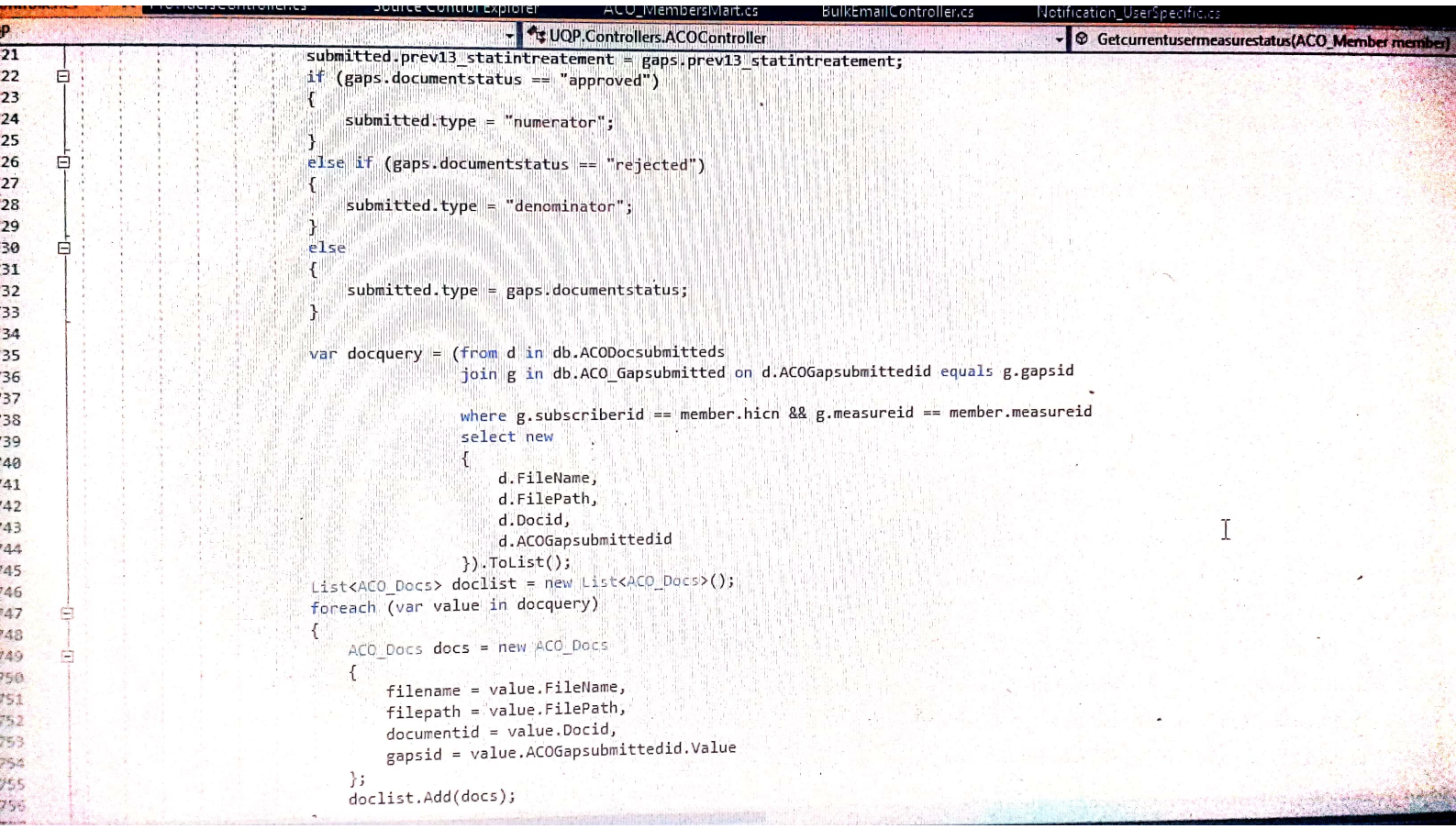


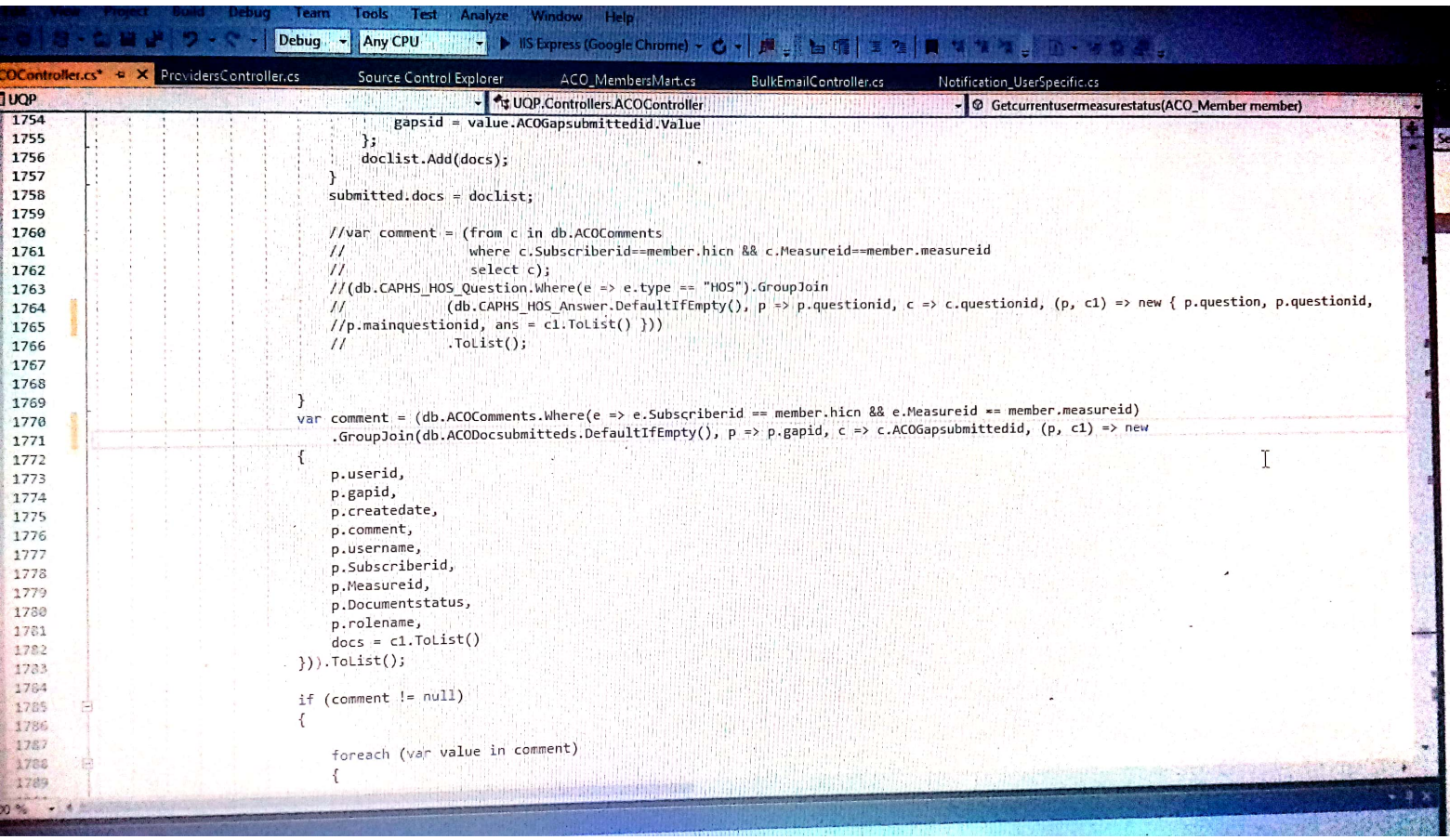


```
submitted.sourcetype = "byupload";
submitted.gapsid = gaps.gapsid;
submitted.h
submitted.s                                                    stname;
submitted.subscriberlastname = gaps.subscriberlastname;
submitted.measureid = gaps.measureid;
submitted.submissiondate = gaps.submissiondate;
submitted.submittedby = gaps.submittedby;
submitted.submittedfirstname = gaps.submittedfirstname;
submitted.submittedlastname = gaps.submittedlastname;
submitted.providerfirstname = gaps.providerfirstname;
submitted.providerlastname = gaps.providerlastname;
submitted.providernpi = gaps.npi;
submitted.auditby = gaps.auditby;
submitted.auditdate = gaps.auditdate;
submitted.auditfirstname = gaps.auditfirstname;
submitted.auditlastname = gaps.auditlastname;
submitted.othercmsreason = gaps.othercmsreason;
submitted.medicalrecordfound = gaps.medicalrecordfound;
submitted.care2_qualifiedforsample = gaps.care2_qualifiedforsample;
submitted.care2_qualifiedforsamplereason = gaps.care2_qualifiedforsamplereason;
submitted.care2_qualifiedformeasure = gaps.care2_qualifiedformeasure;
submitted.care2_qualifiedformeasurereason = gaps.care2_qualifiedformeasurereason;
submitted.care2_patientscreened = gaps.care2_patientscreened;
submitted.dm2_patientdocumentedhistory = gaps.dm2_patientdocumentedhistory;
submitted.dm2_HbA1Cperformed = gaps.dm2_HbA1Cperformed;
submitted.dm2_hba1cdate = gaps.dm2_hba1cdate;
submitted.bcs_qualifiedforsample = gaps.bcs_qualifiedforsample;
submitted.bcs_qualifiedforsamplereason = gaps.bcs_qualifiedforsamplereason;
submitted.bcs_mamogramcompleted = gaps.bcs_mamogramcompleted;
submitted.cc_qualifiedforsample = gaps.cc_qualifiedforsample;
submitted.cc_qualifiedforsamplereason = gaps.cc_qualifiedforsamplereason;
submitted.cc_qualifiedforsamplereason2 = gaps.cc_qualifiedforsamplereason2;
submitted.cc_currentscreening = gaps.cc_currentscreening;
submitted.flu_qualifiedforsample = gaps.flu_qualifiedforsample;
submitted.flu_receivedflushot = gaps.flu_receivedflushot;
```

long ACO_Gapsubmitted.gapsid { get; set; }

```
UQP.Controllers.ACOController                    Getcurrentusermeasurestatus(ACO_Member member)

submitted.cc_currentscreening = gaps.cc_currentscreening;
submitted.flu_qualifiedforsample = gaps.flu_qualifiedforsample;
submitted.flu_receivedflushot = gaps.flu_receivedflushot;
submitted.htn_hypertensiondiagnosis = gaps.htn_hypertensiondiagnosis;
submitted.htn_hypertensiondiagnosisreason = gaps.htn_hypertensiondiagnosisreason;
submitted.htn_bpdocumented = gaps.htn_bpdocumented;
submitted.htn_bpdate = gaps.htn_bpdate;
submitted.htn_bpsystolic = gaps.htn_bpsystolic;
submitted.htn_bpdystolic = gaps.htn_bpdystolic;
submitted.mh1_majordepressiondiagnosis = gaps.mh1_majordepressiondiagnosis;
submitted.mh1_majordepressiondiagnosisreason = gaps.mh1_majordepressiondiagnosisreason;
submitted.mh1_phq9completed = gaps.mh1_phq9completed;
submitted.mh1_phq9greater = gaps.mh1_phq9greater;
submitted.mh1_phq9greaterdate = gaps.mh1_phq9greaterdate;
submitted.mh1_phq9greaterscore = gaps.mh1_phq9greaterscore;
submitted.mh1_phq9measurementperiodcompleted = gaps.mh1_phq9measurementperiodcompleted;
submitted.mh1_phq9measurementperiodcompleteddate = gaps.mh1_phq9measurementperiodcompleteddate;
submitted.mh1_phq9less = gaps.mh1_phq9less;
submitted.mh1_phq9lessdate = gaps.mh1_phq9lessdate;
submitted.mh1_phq9lessscore = gaps.mh1_phq9lessscore;
submitted.prev12_qualified = gaps.prev12_qualified;
submitted.prev12_depressionscreened = gaps.prev12_depressionscreened;
submitted.prev12_positivefordepression = gaps.prev12_positivefordepression;
submitted.prev12_followup = gaps.prev12_followup;
submitted.tobacco_qualified = gaps.tobacco_qualified;
submitted.tobacco_screened = gaps.tobacco_screened;
submitted.tobacco_recentscreening = gaps.tobacco_recentscreening;
submitted.tobacco_cessatinintervention = gaps.tobacco_cessatinintervention;
submitted.documentstatus = gaps.documentstatus;
submitted.prev13_LDLC = gaps.prev13_LDLC;
submitted.prev13_medreason = gaps.prev13_medreason;
submitted.prev13_qualifiedsample = gaps.prev13_qualifiedsample;
submitted.prev13_statintherapy = gaps.prev13_statintherapy;
submitted.prev13_statintreatement = gaps.prev13_statintreatement;
if (gaps.documentstatus == "approved")
{
```

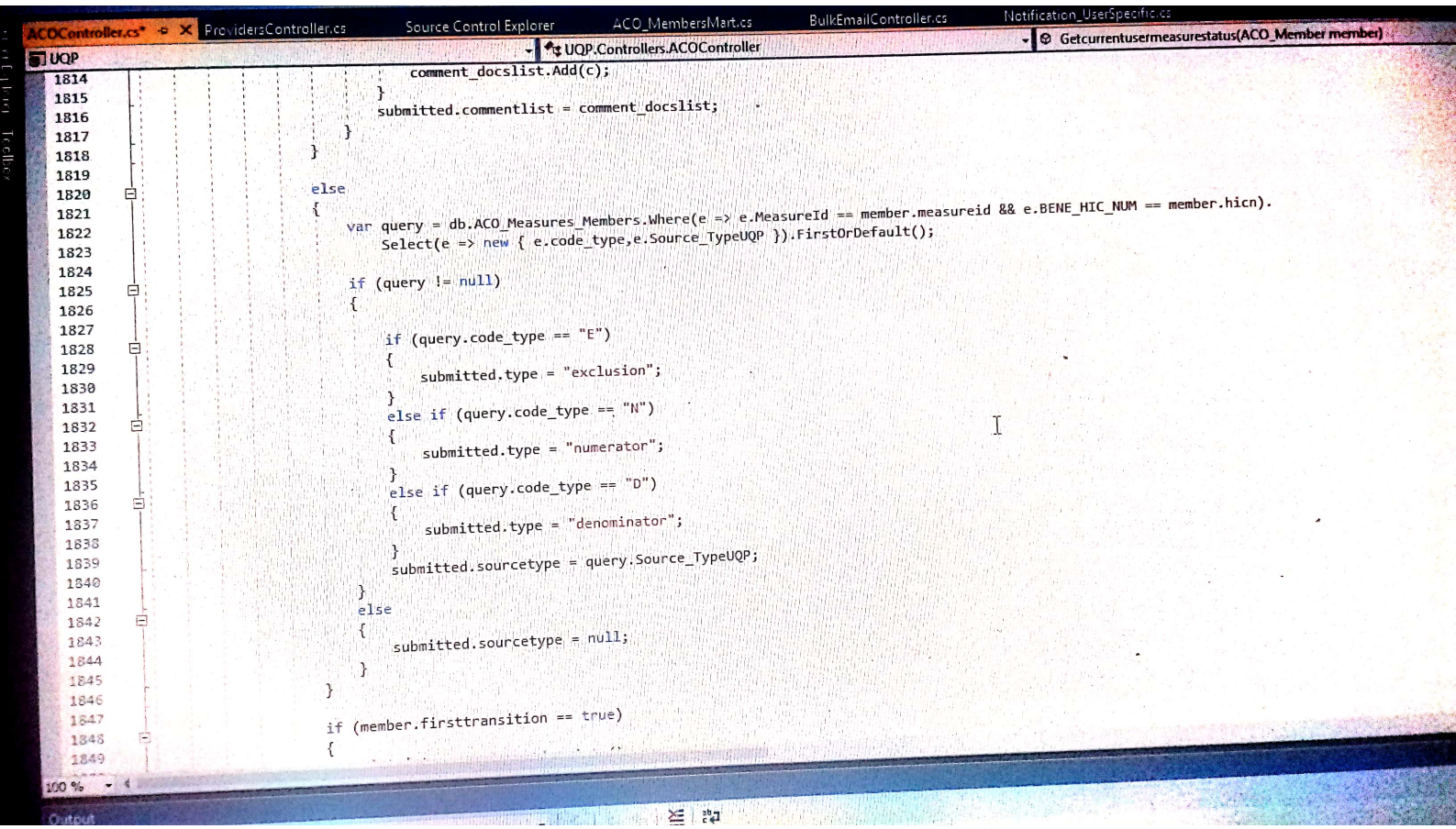


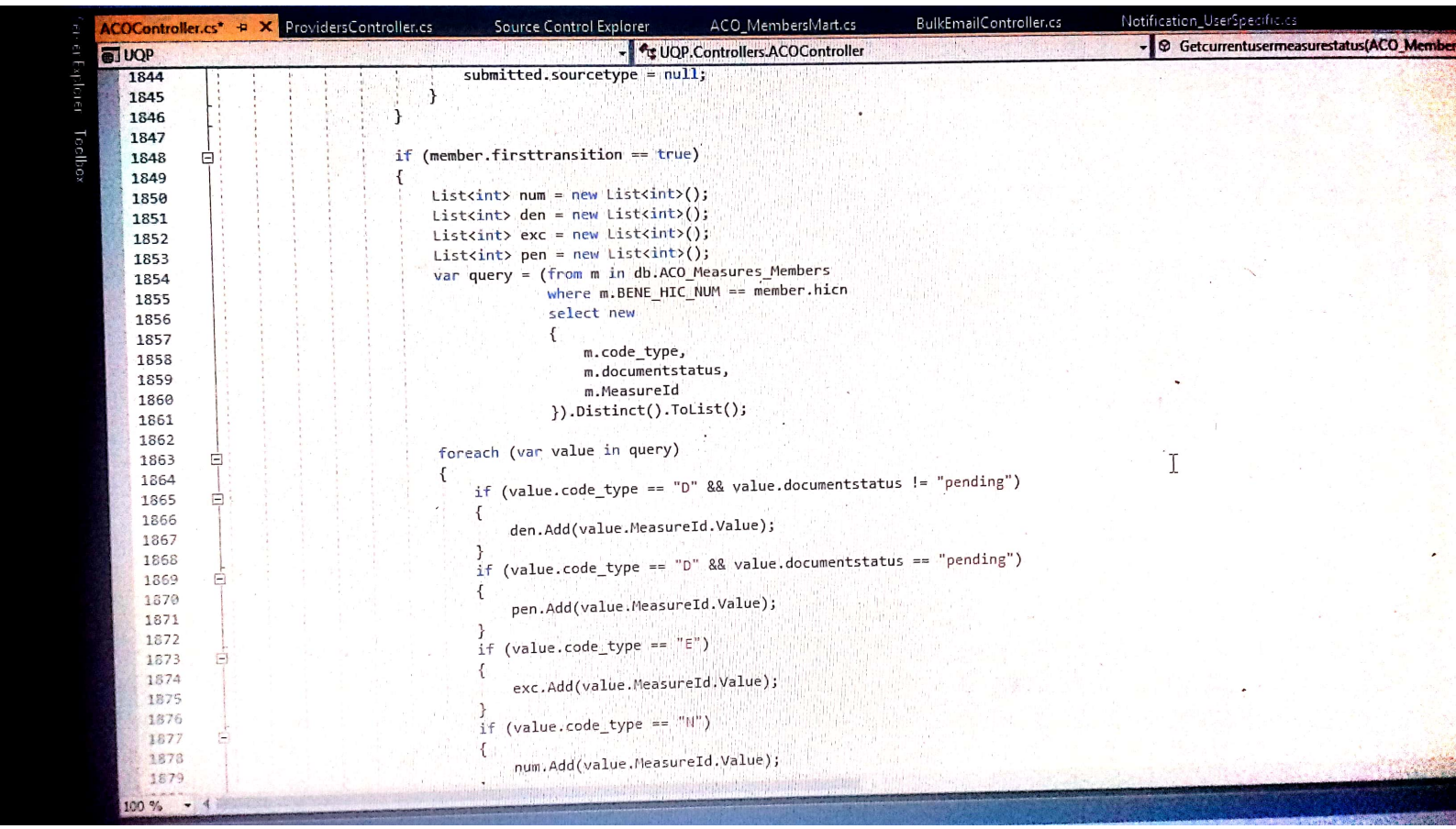


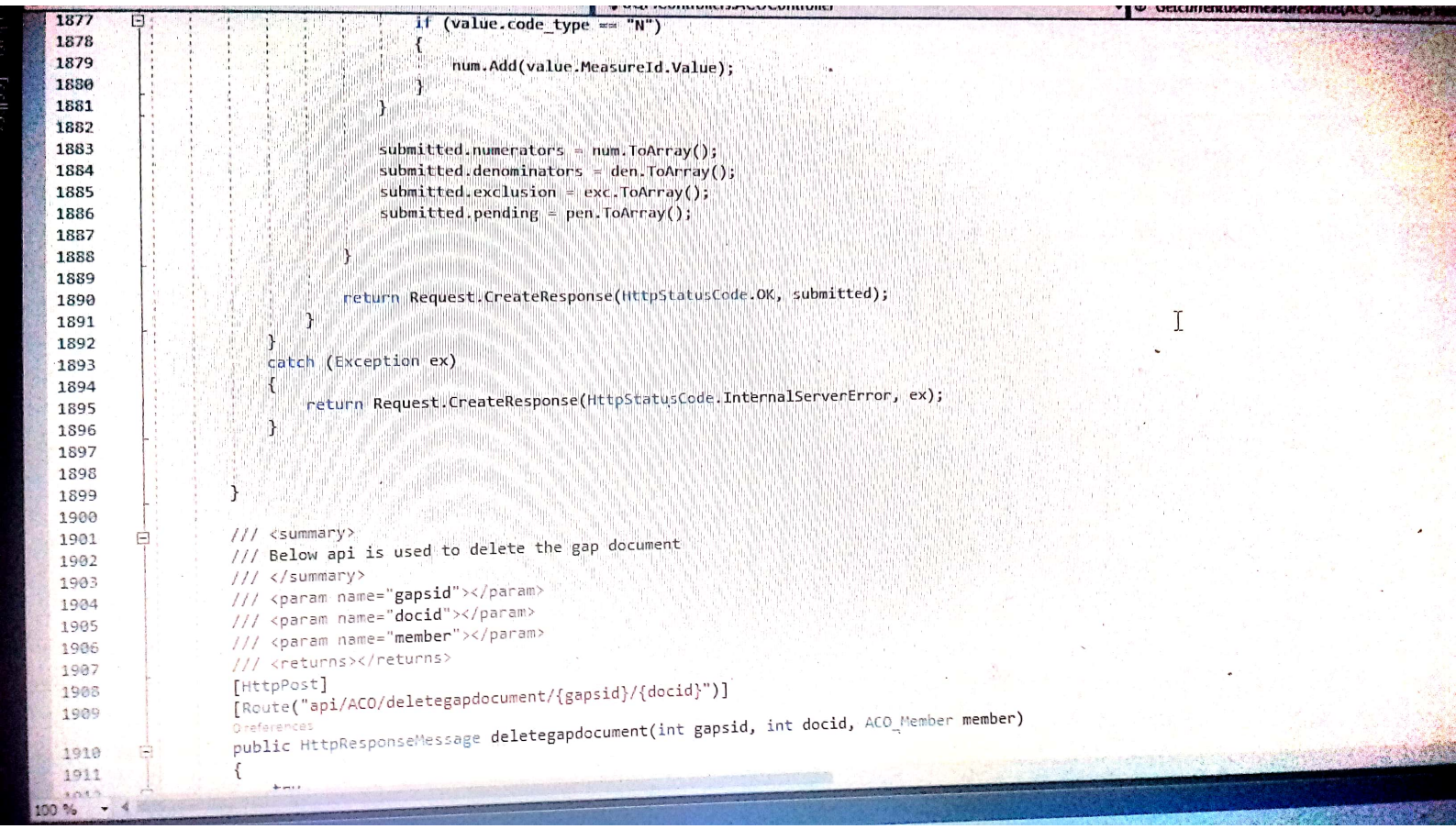
```
1784
1785            if (comment != null)
1786            {
1787
1788                foreach (var value in comment)
1789                {
1790
1791                    ACO_Comment_Docs c = new ACO_Comment_Docs();
1792                    List<ACO_Docs> documentlist = new List<ACO_Docs>();
1793                    c.userid = value.userid;
1794                    c.gapid = value.gapid;
1795                    c.createdate = value.createdate;
1796                    c.comment = value.comment;
1797                    c.username = value.username;
1798                    c.Subscriberid = value.Subscriberid;
1799                    c.Measureid = value.Measureid;
1800                    c.Documentstatus = value.Documentstatus;
1801                    c.rolename = value.rolename;
1802                    foreach (var doc in value.docs)
1803                    {
1804                        ACO_Docs document = new ACO_Docs
1805                        {
1806                            filename = doc.FileName,
1807                            filepath = doc.FilePath,
1808                            documentid = doc.Docid,
1809                            gapsid = doc.ACOGapsubmittedid.Value
1810                        };
1811                        documentlist.Add(document);
1812                    }
1813                    c.documentlist = documentlist;
1814                    comment_docslist.Add(c);
1815                }
1816                submitted.commentlist = comment_docslist;
1817            }
1818        }
1819
```

100 %

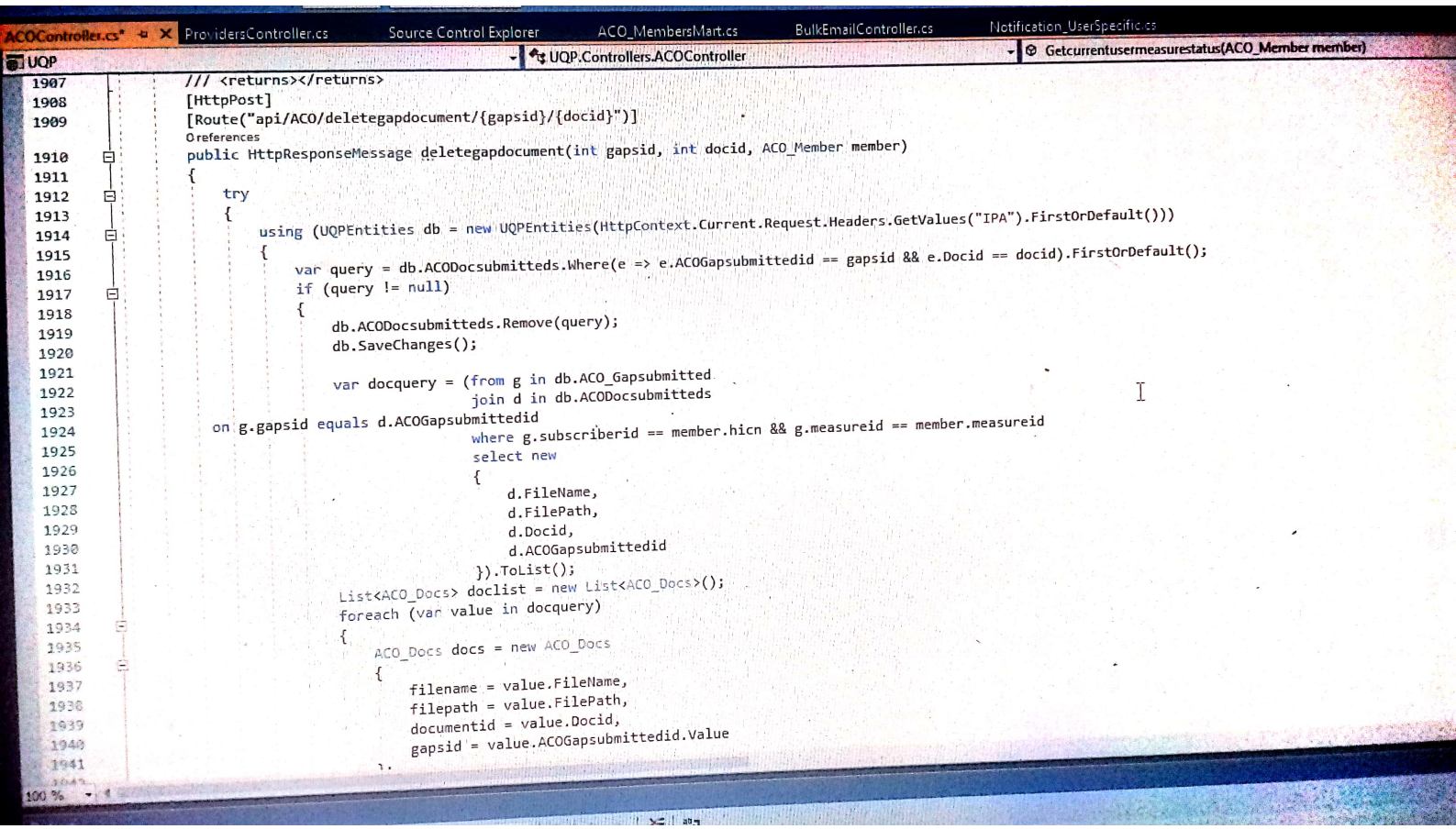
Output

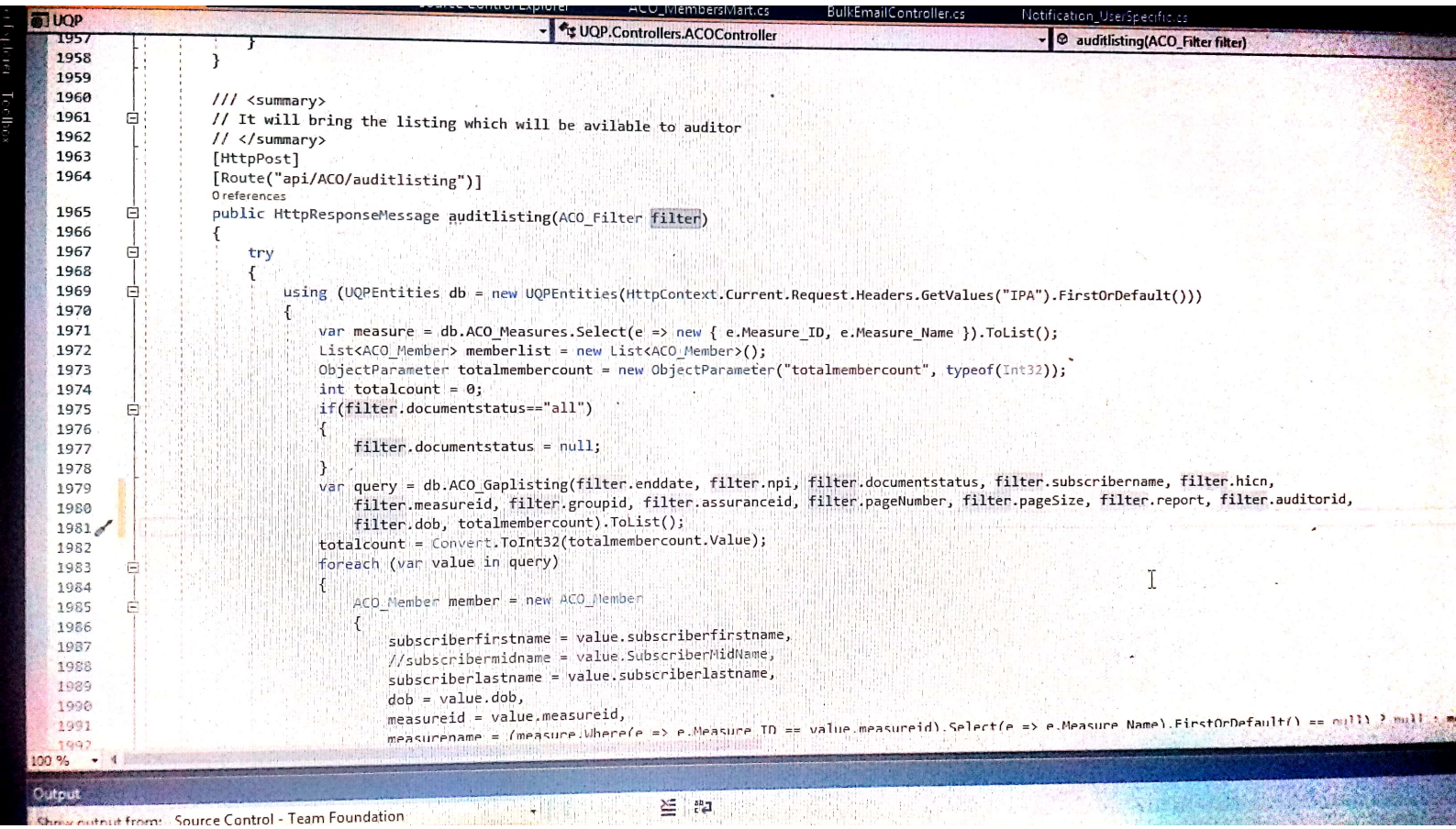


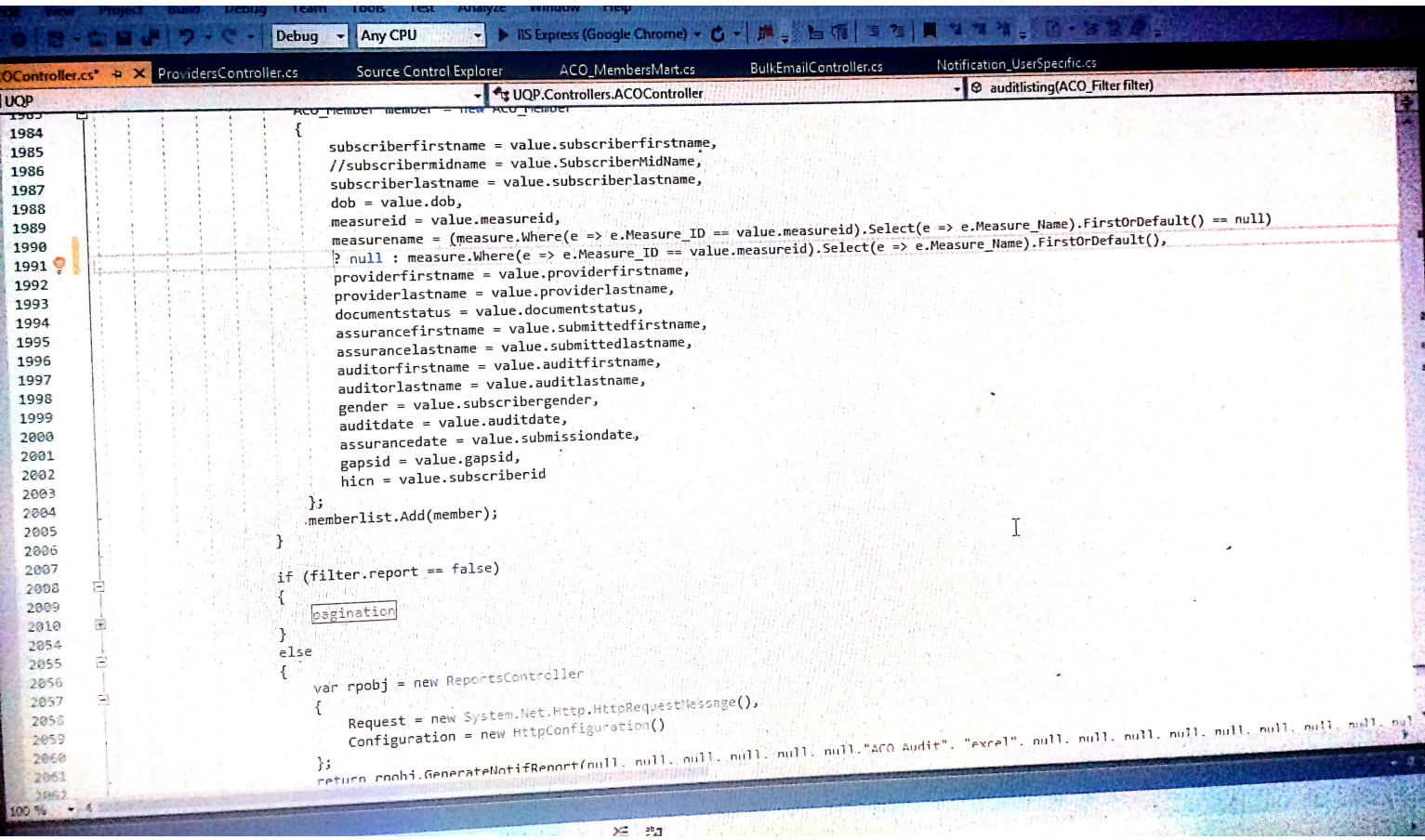


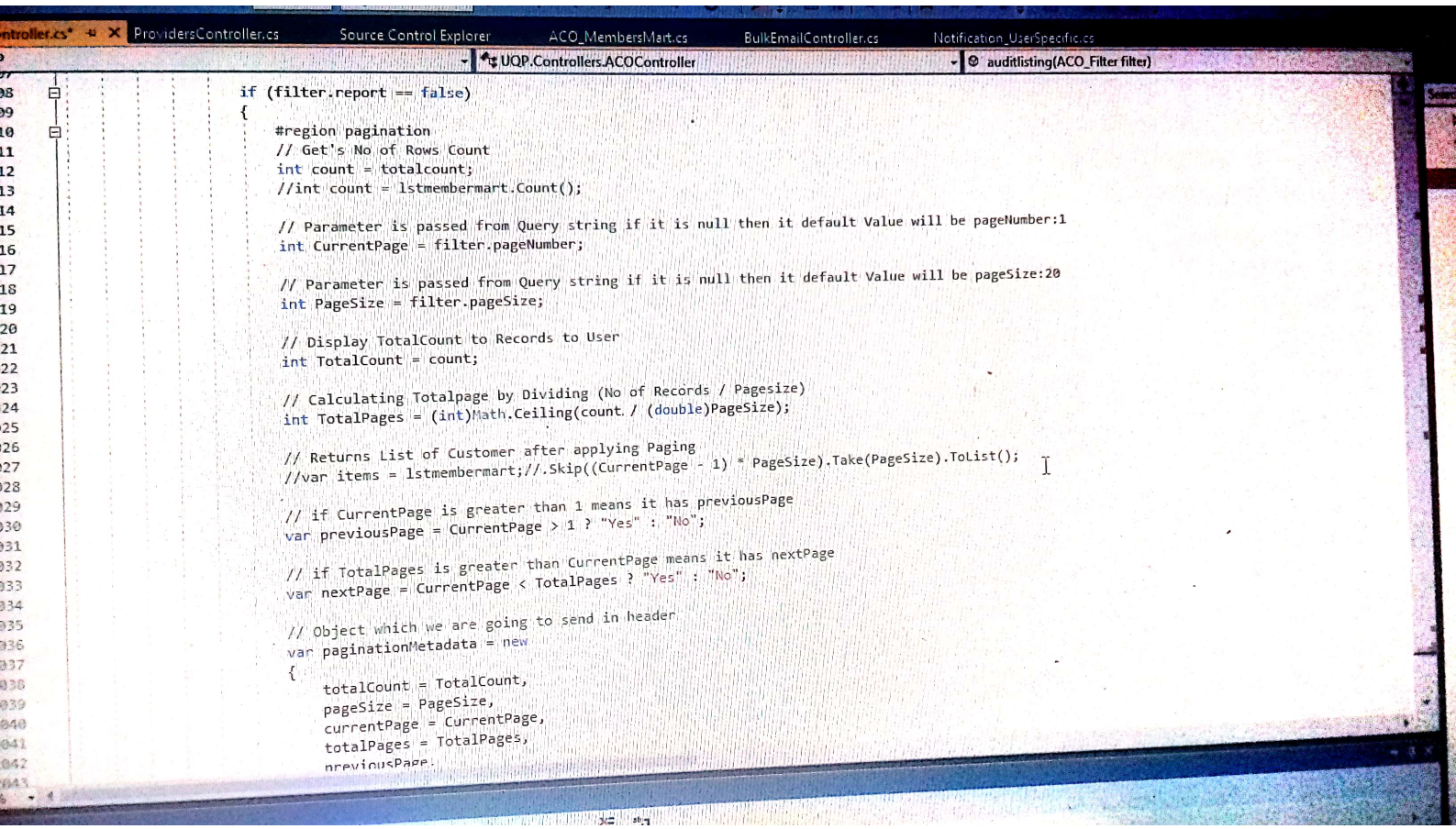


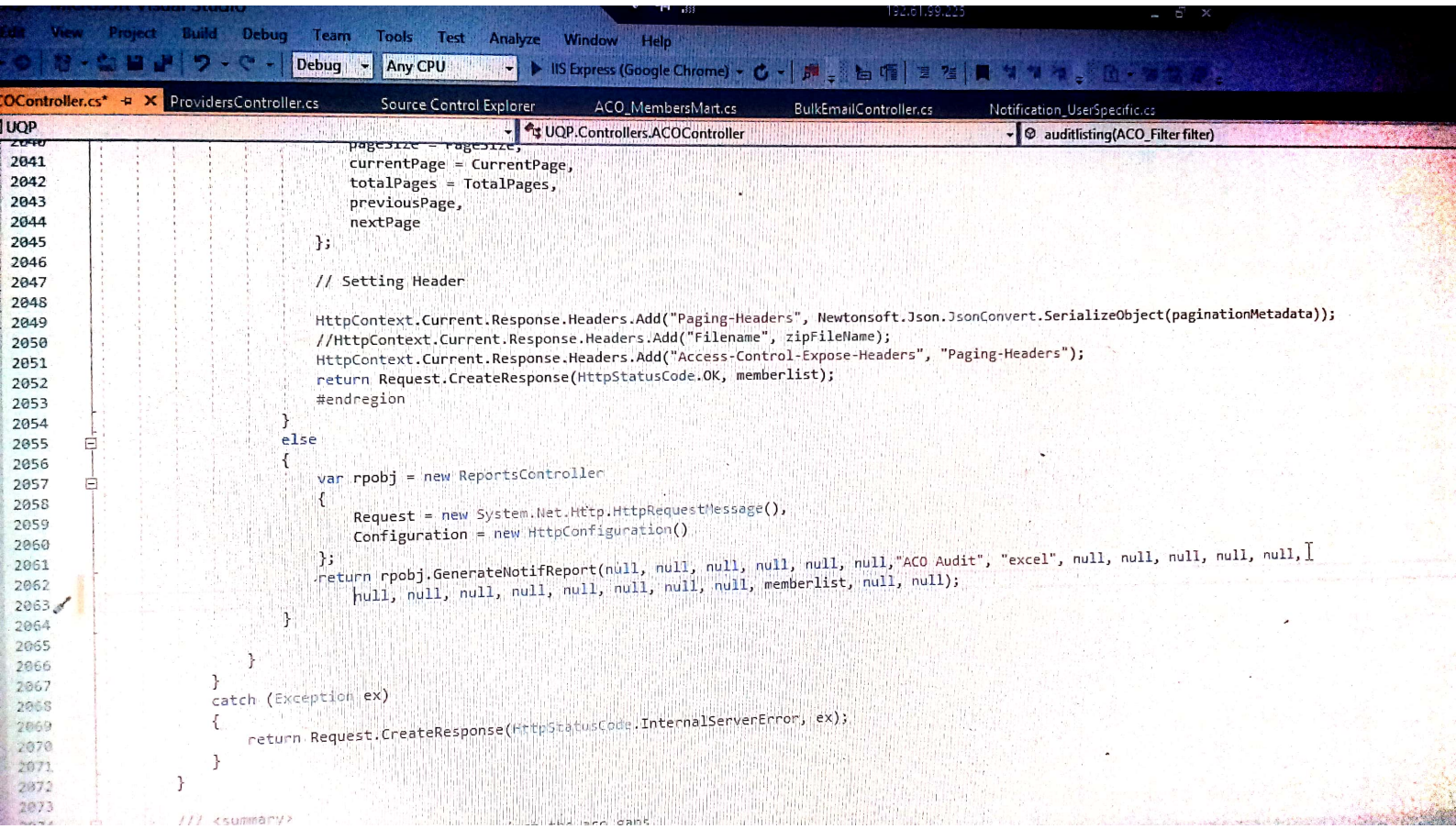
```
                    if (value.code_type == "N")
                    {
                        num.Add(value.MeasureId.Value);
                    }
                }

                submitted.numerators = num.ToArray();
                submitted.denominators = den.ToArray();
                submitted.exclusion = exc.ToArray();
                submitted.pending = pen.ToArray();

            }

            return Request.CreateResponse(HttpStatusCode.OK, submitted);
        }
    }
    catch (Exception ex)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError, ex);
    }


}

/// <summary>
/// Below api is used to delete the gap document
/// </summary>
/// <param name="gapsid"></param>
/// <param name="docid"></param>
/// <param name="member"></param>
/// <returns></returns>
[HttpPost]
[Route("api/ACO/deletegapdocument/{gapsid}/{docid}")]
0 references
public HttpResponseMessage deletegapdocument(int gapsid, int docid, ACO_Member member)
{
    try
```

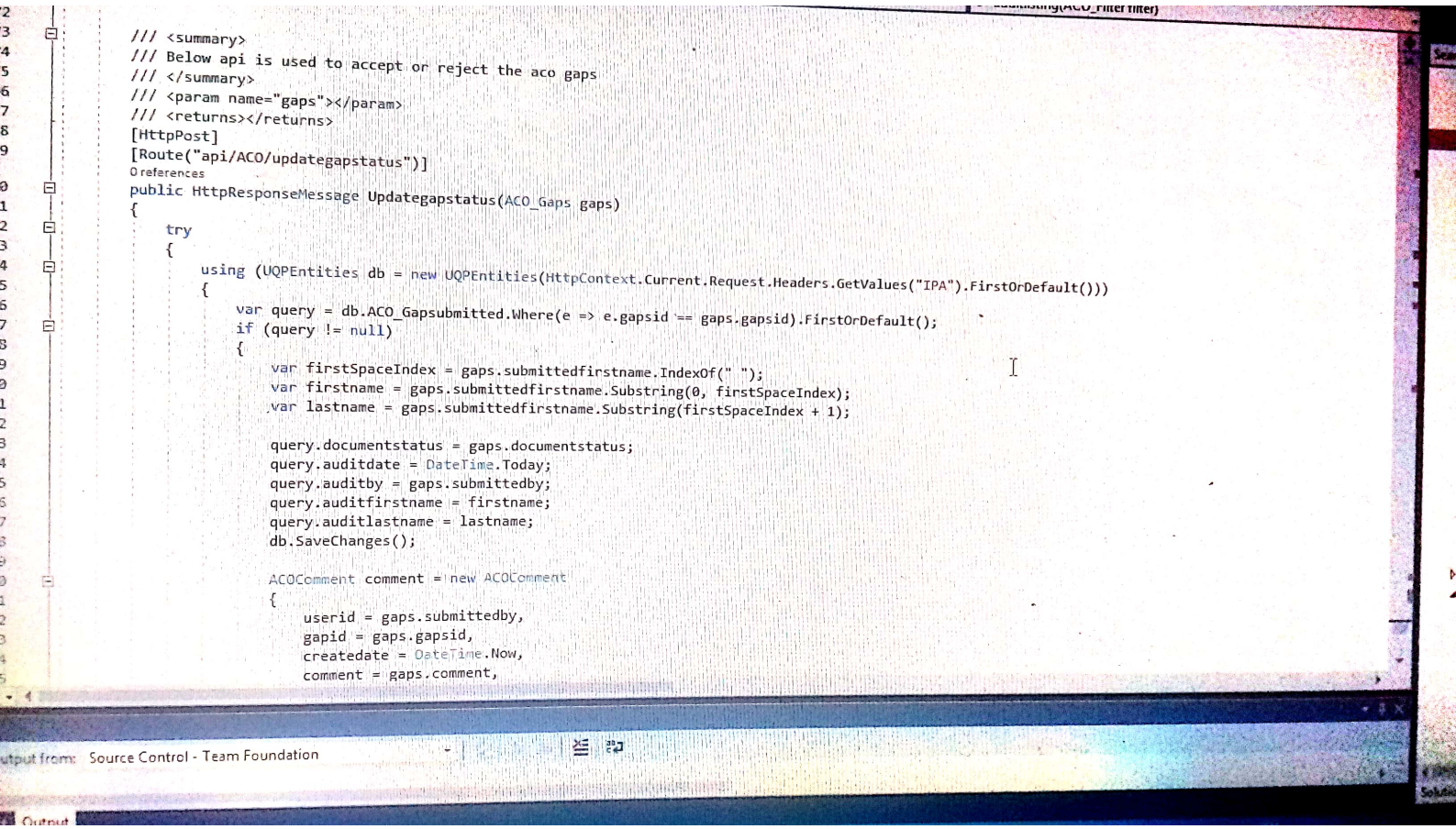












```
/// <summary>
/// Below api is used to accept or reject the aco gaps
/// </summary>
/// <param name="gaps"></param>
/// <returns></returns>
[HttpPost]
[Route("api/ACO/updategapstatus")]
0 references
public HttpResponseMessage Updategapstatus(ACO_Gaps gaps)
{
    try
    {
        using (UQPEntities db = new UQPEntities(HttpContext.Current.Request.Headers.GetValues("IPA").FirstOrDefault()))
        {
            var query = db.ACO_Gapsubmitted.Where(e => e.gapsid == gaps.gapsid).FirstOrDefault();
            if (query != null)
            {
                var firstSpaceIndex = gaps.submittedfirstname.IndexOf(" ");
                var firstname = gaps.submittedfirstname.Substring(0, firstSpaceIndex);
                var lastname = gaps.submittedfirstname.Substring(firstSpaceIndex + 1);

                query.documentstatus = gaps.documentstatus;
                query.auditdate = DateTime.Today;
                query.auditby = gaps.submittedby;
                query.auditfirstname = firstname;
                query.auditlastname = lastname;
                db.SaveChanges();

                ACOComment comment = new ACOComment
                {
                    userid = gaps.submittedby,
                    gapid = gaps.gapsid,
                    createdate = DateTime.Now,
                    comment = gaps.comment,
```

Output from: Source Control - Team Foundation

Output