详细设计说明书

《北师小鸦 UWP》

# 导言

本章对该文档的目的、功能范围、术语、相关文档、参考资料、版本更新进行说明。

## 目的

本文档的目的旨在推动软件工程的规范化，使设计人员遵循统一的详细设计书写规范，节省制作文档的时间，降低系统实现的风险，做到系统设计资料的规范性与全面性，以利于系统的实现、测试、维护、版本升级等。详细设计的详细程度，应达到可以编写程序的程度。

## 范围

本文档用于软件设计阶段的详细设计，它的上游(依据的基线)是《概要设计说明书》，它的下游是源程序清单及单元测试计划，并为单元测试报告提供测试的依据。该范围应覆盖《概要设计说明书》中的功能点列表、性能点列表、接口列表。

软件详细设计的范围是:各子系统的公用模块实现设计、专用模块实现设计、存储过程实现设计、触发器实现设计、外部接口实现设计、部门角色授权设计、其他详细设计等。按照 3 层结构(B/A/S)的布局，详细设计应从下面 3 个方面进行。数据库服务器上的面向数据的设计:数据字典物理设计、基本表物理设计、中间表物理设计(报表设计)、临时表物理设计、视图物理设计、存储过程物理设计、触发器物理设计。应用服务器上的面向业务逻辑的设计：接口数据设计、中间件设计、数据通信传输设计、可视构件设计、非可视构件设计、角色授权设计、功能点设计(功能点列表设计)。浏览器上的面向对象的设计：录入修改界面设计、浏览查询界面设计、登录注册界面设计、信息发布界面设计。

## 术语定义

术语定义，如表 6-16 所示。

|  |  |  |
| --- | --- | --- |
| 序号 | 术语名称 | 术语定义 |
| 1 | 详细设计 | 在概要设计的基础上，对其功能模块或部件进行实现设计，使编程人员 据此能顺利书写出程序代码。 |
| 2 | 存储过程 | 存放在数据库服务器上的一段程序，它能被其他程序调用，以完成对数 据库表的某些规定操作。 |
| 3 | 触发器 | 存放在数据库服务器上的一段程序，当触发条件满足时它就被执行，以完成对数据库表的某些规定操作。 |
| 4 | 算法 | 详细设计中实现某项功能的数据处理方法及处理流程。 |
| 5 | PC | 个人计算机 |
| 6 | Surface | 微软 Surface 平板电脑 |
| 7 | Xbox | 微软Xbox游戏机 |
| 8 | WP | 微软 Windows 手机  Windows Phone |
| 9 | C# | 微软 C#/C Sharp 编程语言 |
| 10 | Windows | 微软视窗操作系统 |
| 11 | C/S | 客户/服务架构 Browser/Server |
| 12 | GPA | 平均绩点 Grade Point Average |

## 参考资料

[1] 《概要设计说明书》

[2] 《需求分析说明书》

[3] 《软件合同》

[4] 命名规范

[5] 程序设计规范

[6] 界面设计规范

## 相关文档

[1] 源程序清单

[2] 单元测试计划及报告

[3] 《用户使用手册》

# 模块实现设计

## 公用模块设计

### 解析器助手模块

模块编号：G-001

模块名称：ParserHelper

模块功能：

1. 获取首个元素文本
2. 获取首个元素

模块背景描述：

解析HTML时，经常获取指定Tag的元素列表，需要得到其中匹配的一个元素。如果未匹配，则返回空。

模块算法设计：

/// <summary>

        /// Get text of first element from a element collection

        /// </summary>

        /// <param name="elements">Collection of element</param>

        /// <returns><see cref="string"/>. Returns <c>null</c> if there is no element.</returns>

        public static string GetFirstElementText(IHtmlCollection<IElement> elements)

        {

            if (elements.Count() > 0)

            {

                return elements[0].TextContent;

            }

            else

            {

                return null;

            }

        }

        /// <summary>

        /// Get first element from a element collection

        /// </summary>

        /// <param name="elements">Collection of element</param>

        /// <returns><see cref="IElement"/>. Returns <c>null</c> if there is no element.</returns>

        public static IElement GetFirstElement(IHtmlCollection<IElement> elements)

        {

            if (elements.Count() > 0)

            {

                return elements[0];

            }

            else

            {

                return null;

            }

        }

    }

模块编写者：许宏旭

编写日期：2017.4.22

模块修订者：许宏旭

修订日期：2017.4.23

模块测试者：许宏旭

测试日期：2017.4.23

## 专用模块设计

### 统一身份认证登录

模块编号：M0-1

模块名称：Login

模块背景描述：登录统一身份认证系统，以进入教务系统

模块算法设计：

/// <summary>

        /// Get login parameters from the login page

        /// </summary>

        /// <returns>A 2-element array of <see cref="string"/>: [0] => lt, [1] => execution</returns>

        private async Task<string[]> FetchLoginParams()

        {

            string[] retParams = new string[2];                 // there'll be 2 params

            var res = await m\_Session.Req                       // GET URL\_LOGIN with specific UA

               .Url(URL\_LOGIN)

               .ClearCookie()

               .Header(HEADER\_USER\_AGENT, USER\_AGENT)

               .Get();

            string body = await res.Content();                  // fetch the response body (html)

            // extract the value of param "lt"

            Match mc = Regex.Match(body, "input type=\"hidden\" name=\"lt\" value=\"(.\*)\"");

            retParams[0] = mc.Groups[1].Value;

            // extract the value of param "execution"

            mc = Regex.Match(body, "input type=\"hidden\" name=\"execution\" value=\"(.\*)\"");

            retParams[1] = mc.Groups[1].Value;

            return retParams;

        }

/// <summary>

        /// Login to the BNU universal authentication platform

        /// </summary>

        /// <returns>Error message. Returns <c>null</c> if succeed.</returns>

        public async Task<string> Login()

        {

            if (m\_IsLogined) return null;

            // Fetch login params needed

            string[] loginParams = await FetchLoginParams();

            var res = await m\_Session.Req

                .Url(URL\_LOGIN)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Data("username", m\_Username)

                .Data("password", m\_Password)

                .Data("code", "code")

                .Data("lt", loginParams[0])

                .Data("execution", loginParams[1])

                .Data("\_eventId", "submit")

                .Post();

            // Decode html body by GBK

            string body = "";

            body = res.Content("GBK").Result;

            var doc = m\_Parser.Parse(body);

            // Init error message

            string error = "登录失败";

            // If no "KINGOSOFT高校数字校园综合管理平台" found, then there will be errors

            if (!body.Contains("KINGOSOFT高校数字校园综合管理平台"))

            {

                // Get error message element: <span id="error\_message\_show">

                var msg = doc.GetElementById("error\_message\_show");

                // Element found, then assign error message

                if (msg != null)

                {

                    error = msg.TextContent;

                }

                return error;

            }

            // Otherwise, logined successfully.

            m\_IsLogined = true;

            return null;

        }

模块编写者：许宏旭

编写日期：2017.4.22

模块修订者：许宏旭

修订日期：2017.4.22

模块测试者：许宏旭

测试日期：2017.4.22

### 获取学生信息模块

模块编号：M0-2

模块名称：Get Student Info

模块背景描述：数据统计以及后续查询操作需要学号、年级等信息

模块算法设计：

/// <summary>

        /// Get student info

        /// <list type="bullet">

        ///     <item>student\_id: xh</item>

        ///     <item>grade: nj</item>

        ///     <item>major: zymc</item>

        ///     <item>major\_id: zydm</item>

        ///     <item>school\_year: xn</item>

        ///     <item>semester: xq\_m</item>

        /// </list>

        /// </summary>

        /// <returns><see cref="StudentInfo"/><returns>

        public async Task<StudentInfo> GetStudentInfo()

        {

            if (m\_StudentInfo != null)

            {

                return m\_StudentInfo;

            }

            var res = await m\_Session.Req

                .Url(URL\_STUDENT\_INFO)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Header(HEADER\_REFERER, REFERER)

                .Post();

            string body = await res.Content();

            // Update login state to avoid session expiration

            if (!UpdateLoginState(body))

                return null;

            // Parse the fetched XML body

            var xmlParser = new XmlParser();

            var doc       = xmlParser.Parse(body);

            string studentId, grade, major, majorId, schoolYear, semester;

            studentId  = ParserHelper.GetFirstElementText(doc.GetElementsByTagName("xh"));

            grade      = ParserHelper.GetFirstElementText(doc.GetElementsByTagName("nj"));

            major      = ParserHelper.GetFirstElementText(doc.GetElementsByTagName("zymc"));

            majorId    = ParserHelper.GetFirstElementText(doc.GetElementsByTagName("zydm"));

            schoolYear = ParserHelper.GetFirstElementText(doc.GetElementsByTagName("xn"));

            semester   = ParserHelper.GetFirstElementText(doc.GetElementsByTagName("xq\_m"));

            // If no major or grade info

            if (majorId == null || grade == null || majorId == "" || grade == "")

            {

                // Fetch them by another way

                GradeInfo gradeInfo = await GetGradeInfo(studentId);

                major = gradeInfo.Major;

                majorId = gradeInfo.MajorId;

                grade = gradeInfo.Grade;

            }

            m\_StudentInfo = new StudentInfo(studentId, grade, major, majorId, schoolYear, semester);

            return m\_StudentInfo;

        }

        /// <summary>

        /// Get grade and major info

        /// </summary>

        /// <param name="studentId">Student Id</param>

        /// <returns><see cref="GradeInfo"/></returns>

        public async Task<GradeInfo> GetGradeInfo(string studentId)

        {

            var res = await m\_Session.Req

                .Url(URL\_GRADE\_INFO)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Header(HEADER\_REFERER, REFERER)

                .Data("xh", studentId)

                .Post();

            string body = await res.Content("UTF-8");

            if (!UpdateLoginState(body))

                return null;

            RequestResult result = JsonConvert.DeserializeObject<RequestResult>(body);

            return new GradeInfo(JsonConvert.DeserializeObject<GradeInfo.\_GradeInfo>(result.Result));

        }

        /// <summary>

        /// Get student details

        /// </summary>

        /// <returns><see cref="StudentDetails"/></returns>

        public async Task<StudentDetails> GetStudentDetails()

        {

            var res = await m\_Session.Req

                .Url(URL\_STUDENT\_DETAILS)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Header(HEADER\_REFERER, REFERER)

                .Post();

            string body = await res.Content();

            if (!UpdateLoginState(body))

                return null;

            var xmlParser = new XmlParser();

            var doc       = xmlParser.Parse(body);

            var info = ParserHelper.GetFirstElement(doc.GetElementsByTagName("info"));

            if (info == null)

            {

                return null;

            }

            StudentDetails details = new StudentDetails(

                address:            ParserHelper.GetFirstElementText(info.GetElementsByTagName("txdz")),

                avatarId:           ParserHelper.GetFirstElementText(info.GetElementsByTagName("zpid")),

                birthday:           ParserHelper.GetFirstElementText(info.GetElementsByTagName("csrq")),

                className:          ParserHelper.GetFirstElementText(info.GetElementsByTagName("bjmc")),

                college:            ParserHelper.GetFirstElementText(info.GetElementsByTagName("yxb")),

                collegeWill:        ParserHelper.GetFirstElementText(info.GetElementsByTagName("zymc")),

                cultureStandard:    ParserHelper.GetFirstElementText(info.GetElementsByTagName("whcd")),

                educationLevel:     ParserHelper.GetFirstElementText(info.GetElementsByTagName("pycc")),

                email:              ParserHelper.GetFirstElementText(info.GetElementsByTagName("dzyx")),

                gaokaoId:           ParserHelper.GetFirstElementText(info.GetElementsByTagName("gkksh")),

                gender:             ParserHelper.GetFirstElementText(info.GetElementsByTagName("xb")),

                id:                 ParserHelper.GetFirstElementText(info.GetElementsByTagName("yhxh")),

                idNumber:           ParserHelper.GetFirstElementText(info.GetElementsByTagName("sfzjh")),

                middleSchool:       ParserHelper.GetFirstElementText(info.GetElementsByTagName("sydw")),

                mobile:             ParserHelper.GetFirstElementText(info.GetElementsByTagName("dh")),

                name:               ParserHelper.GetFirstElementText(info.GetElementsByTagName("xm")),

                nationality:        ParserHelper.GetFirstElementText(info.GetElementsByTagName("mz")),

                number:             ParserHelper.GetFirstElementText(info.GetElementsByTagName("xh")),

                pinyin:             ParserHelper.GetFirstElementText(info.GetElementsByTagName("xmpy")),

                registrationGrade:  ParserHelper.GetFirstElementText(info.GetElementsByTagName("rxnj")),

                registrationTime:   ParserHelper.GetFirstElementText(info.GetElementsByTagName("bdtime")),

                schoolSystem:       ParserHelper.GetFirstElementText(info.GetElementsByTagName("xz")),

                speciality:         ParserHelper.GetFirstElementText(info.GetElementsByTagName("lqzy"))

            );

            return details;

        }

模块编写者：许宏旭

编写日期：2017.4.23

模块修订者：许宏旭

修订日期：2017.4.23

模块测试者：许宏旭

测试日期：2017.4.23

### 获取考试成绩模块

模块编号：M1-2

模块名称：Get Exam Score

模块背景描述：获取考试轮次和指定轮次成绩的后端抓取功能逻辑

模块算法设计：

/// <summary>

        /// Get exam rounds

        /// </summary>

        /// <returns>A list of <see cref="ExamRound"/></returns>

        public async Task<List<ExamRound>> GetExamRounds()

        {

            var res = await m\_Session.Req

                .Url(URL\_DROPLIST)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Header(HEADER\_REFERER, REFERER)

                .Data("comboBoxName", DROP\_EXAM\_NAME)

                .Post();

            string body = await res.Content("UTF-8");

            if (!UpdateLoginState(body))

                return null;

            var list = JsonConvert.DeserializeObject<List<ExamRound>>(body);

            return list.OrderByDescending(o => o.Code).ToList();

        }

        /// <summary>

        /// Get exam scores

        /// </summary>

        /// <param name="year">Specific which year to query. If 0 is given, all scores will be returned.</param>

        /// <param name="term">Specific which term to query.</param>

        /// <param name="isOnlyMajor">Specific whether scores of minor profession will be returned.</param>

        /// <returns>A list of <see cref="ExamScore"/></returns>

        public async Task<List<ExamScore>> GetExamScores(int year, int term, bool isOnlyMajor)

        {

            var req = m\_Session.Req

                .Url(URL\_EXAM\_SCORE)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Header(HEADER\_REFERER, REFERER\_EXAM\_SCORE)

                .Data("ysyx", "yscj")

                .Data("userCode", (await GetStudentInfo()).StudentId)

                .Data("zfx", isOnlyMajor ? "0" : "1")

                .Data("ysyxS", "on")

                .Data("sjxzS", "on")

                .Data("zfxS", "on");

            if (year == 0)

            {

                // If year == 0, Then get all exam scores

                req.Data("sjxz", "sjxz1");

            }

            else

            {

                // Else get scores for specific semester

                req.Data("sjxz", "sjxz3")

                    .Data("xn", year.ToString())

                    .Data("xn1", (year + 1).ToString())

                    .Data("xq", term.ToString());

            }

            var res = await req.Post();

            string body = await res.Content("GBK");

            if (!UpdateLoginState(body))

                return null;

            var doc = m\_Parser.Parse(body);

            var scores = new List<ExamScore>();

            if (doc.GetElementsByTagName("tbody").Count() == 0)

            {

                // No result

                return scores;

            }

            var table = doc.GetElementsByTagName("tbody")[0];

            var rows  = table.GetElementsByTagName("tr");

            string lastTerm = "";

            foreach (var tr in rows)

            {

                var cols = tr.GetElementsByTagName("td");

                string currentTerm = cols[0].TextContent.Trim();

                if (currentTerm == "")

                {

                    currentTerm = lastTerm;

                }

                else

                {

                    lastTerm = currentTerm;

                }

                if (cols.Count() < 9) continue;

                scores.Add(new ExamScore(

                        semester:               currentTerm,

                        courseName:             cols[1].TextContent,

                        courseCredit:           cols[2].TextContent,

                        classification:         cols[3].TextContent,

                        score1:                 cols[5].TextContent,

                        score2:                 cols[6].TextContent,

                        score:                  cols[7].TextContent,

                        doLearnForFirstTime:    "初修" == cols[4].TextContent.Trim(),

                        isMajor:                "主修" == cols[8].TextContent.Trim()

                ));

            }

            scores.OrderByDescending(o => o.Score);

            return scores;

        }

        /// <summary>

        /// Get exam scores of all semesters

        /// </summary>

        /// <param name="isOnlyMajor">Specific whether scores of minor profession will be returned.</param>

        /// <returns>A list of <see cref="ExamScore"/></returns>

        public async Task<List<ExamScore>> GetExamScores(bool isOnlyMajor)

        {

            return await GetExamScores(0, 0, isOnlyMajor);

        }

模块编写者：许宏旭

编写日期：2017.4.28

模块修订者：许宏旭

修订日期：2017.4.28

模块测试者：许宏旭

测试日期：2017.4.28

### 获取考试安排模块

模块编号：M1-1

模块名称：Get Exam Arrangement

模块背景描述：获取指定轮次考试安排信息

模块算法设计：

/// <summary>

        /// Get exam arrangement of specific exam round

        /// </summary>

        /// <param name="round"><see cref="ExamRound"/></param>

        /// <returns>A list of <see cref="ExamArrangement"/></returns>

        public async Task<List<ExamArrangement>> GetExamArrangement(ExamRound round)

        {

            var res = await m\_Session.Req

                .Url(URL\_DATA\_TABLE + TABLE\_EXAM\_ARRANGEMENT)

                .Header(HEADER\_USER\_AGENT, USER\_AGENT)

                .Header(HEADER\_REFERER, REFERER)

                .Data("xh", "")

                .Data("xn", round.Year)

                .Data("xq", round.Semester)

                .Data("kslc", round.Round)

                .Data("xnxqkslc", round.Code)

                .Post();

            string body = await res.Content("GBK");

            if (!UpdateLoginState(body))

                return null;

            var doc = m\_Parser.Parse(body);

            var arrangementList = new List<ExamArrangement>();

            for (int i = 0; ; ++i)

            {

                string prefix = "tr" + i + "\_";

                var courseNameEl = doc.GetElementById(prefix + "kc");

                if (courseNameEl == null) {

                    // IF no tr{i}\_kc, THEN no new lines.

                    break;

                }

                ExamArrangement arrangement = new ExamArrangement(

                    courseName:     courseNameEl.TextContent,

                    credit:         doc.GetElementById(prefix + "xf").TextContent,

                    classification: doc.GetElementById(prefix + "lb").TextContent,

                    examType:       doc.GetElementById(prefix + "khfs").TextContent,

                    time:           doc.GetElementById(prefix + "kssj").TextContent,

                    location:       doc.GetElementById(prefix + "ksdd").TextContent,

                    seat:           doc.GetElementById(prefix + "zwh").TextContent

                );

                arrangementList.Add(arrangement);

            }

            arrangementList.Sort((a, b) =>

            {

                if (a.BeginTime.HasValue && a.EndTime.HasValue &&

                    b.BeginTime.HasValue && b.EndTime.HasValue)

                {

                    bool aHasEnded = a.EndTime.Value < DateTime.Now;

                    bool bHasEnded = b.EndTime.Value < DateTime.Now;

                    if (aHasEnded && bHasEnded)

                    {

                        return b.EndTime.Value.CompareTo(a.EndTime.Value);

                    }

                    else if (!aHasEnded && !bHasEnded)

                    {

                        return a.BeginTime.Value.CompareTo(b.BeginTime.Value);

                    }

                    else if (aHasEnded && !bHasEnded)

                    {

                        return 1000;

                    }

                    else

                    {

                        return -1000;

                    }

                }

                return 1;

            });

            return arrangementList;

        }

模块编写者：许宏旭

编写日期：2017.4.29

模块修订者：许宏旭

修订日期：2017.4.29

模块测试者：许宏旭

测试日期：2017.4.29

### 获取课程表模块

模块编号：M2-1

模块名称：Get Table Course

模块背景描述：获取指定学期的课程表课程清单

模块算法设计：

/// <summary>

/// Get table semesters

/// </summary>

/// <returns>A list of <see cref="TableSemester"/></returns>

public async Task<List<TableSemester>> GetTableSemesters()

{

    var res = await m\_Session.Req

        .Url(URL\_DROPLIST)

        .Header(HEADER\_USER\_AGENT, USER\_AGENT)

        .Header(HEADER\_REFERER, REFERER)

        .Data("comboBoxName", DROP\_SEMESTER)

        .Post();

    string body = await res.Content("UTF-8");

    if (!UpdateLoginState(body))

        return null;

    return JsonConvert.DeserializeObject<List<TableSemester>>(body);

}

/// <summary>

/// Get courses of timetable for specific semester

/// </summary>

/// <param name="semester"><see cref="TableSemester"/></param>

/// <returns>A list of <see cref="TableCourse"/></returns>

public async Task<List<TableCourse>> GetTableCourses(TableSemester semester)

{

    // Base64 encoding content

    string content = "xn=" + semester.Year

        + "&xq=" + semester.Semester

        + "&xh=" + (await GetStudentInfo()).StudentId;

    content = Convert.ToBase64String(System.Text.Encoding.ASCII.GetBytes(content));

    var res = await m\_Session.Req

        .Url(URL\_TIMETABLE + content)

        .Header(HEADER\_USER\_AGENT, USER\_AGENT)

        .Header(HEADER\_REFERER, REFERER\_TIMETABLE)

        .Get();

    string body = await res.Content();

    if (!UpdateLoginState(body))

        return null;

    var doc = m\_Parser.Parse(body);

    var courses = new List<TableCourse>();

    var table = ParserHelper.GetFirstElement(doc.GetElementsByTagName("tbody"));

    if (table == null)

    {

        return courses;

    }

    var rows = table.GetElementsByTagName("tr");

    foreach (var tr in rows)

    {

        var cols = tr.GetElementsByTagName("td");

        if (cols.Count() < 14) continue;

        courses.Add(new TableCourse(

            code:           cols[13].TextContent,

            name:           cols[0].TextContent,

            credit:         cols[2].TextContent,

            teacher:        cols[4].TextContent,

            locationTime:   cols[5].TextContent,

            isFreeToListen: cols[8].TextContent == "是"

        ));

    }

    return courses;

}

模块编写者：许宏旭

编写日期：2017.4.30

模块修订者：许宏旭

修订日期：2017.4.30

模块测试者：许宏旭

测试日期：2017.4.30

## 存储过程设计

INSERT INTO Users VALUES (姓名, 学号, 年级, 手机号, Email);

# 接口实现设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 接口名称 | 接口内容 | 接口协议 | 接口数据结构 |
| 1 | 数字京师统一身份认证平台接口 | 京师统一身份认证系统登录 | Oauth | Cookies |
| 2 | 查看课表接口 | 显示本周或指定周课程表 | HTTP | List<TableCourse> |
| 3 | 考试安排接口 | 查询指定学期的考试安排 | HTTP | List<ExamArrangement> |
| 4 | 成绩查询接口 | 按要求查询成绩 | HTTP | List<ExamScore> |
| 5 | 评教接口 | 对指定学期的教师教学进行评价 | HTTP | List<Evaluation> |
| 6 | 自习室查询接口 | 查询指定时间段的自习室 | HTTP | List<Room> |
| 7 | 图书馆馆藏查询接口 | 查询图书馆馆藏 | HTTP | List<LibraryBook> |
| 8 | 图书馆选座接口 | 查询并提交图书馆选座 | HTTP | List<LibrarySeat> |
| 9 | 北师大网关接口 | 登录北师大网关 | HTTP | String(HTML) |

# 其他实现设计

## 角色授权设计

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 模块名 | 宣传员 | 公关文创人员 | 开发员 | 维护员 | 客服 | 统计员 |
| 教务系统数据抓取模块 |  |  | √ | √ |  | √ |
| 教务系统数据处理模块 |  |  | √ | √ |  | √ |
| 考试安排模块 | √ | √ | √ |  | √ |  |
| 成绩查询模块 | √ | √ | √ |  | √ |  |
| 课程表模块 | √ | √ | √ |  | √ |  |
| 图书馆管理模块 | √ | √ | √ |  | √ |  |
| 自习室查询模块 | √ | √ | √ |  | √ |  |
| 北师大网关模块 | √ | √ | √ |  | √ |  |

# 详细设计检查列表

## 功能设计检查列表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 功能名称 | 功能描述 | 输入内容 | 系统响应 | 输出内容 | 是否实现 |
| 1 | 京师认证登陆 | 用户输入用户名、密码，登录京师统一身份认证系统 | 用户名 | 登陆京师认证 | 登陆信息  （成功或失败）  用户信息  （Cookies） | 是 |
| 2 | 课程表 | 选择指定学期，导入指定学期课程，显示本周或指定周课程表 | 指定学期 | 获取课程表 | 课程安排列表（课程名称、时间地点、任课教师）  图形化课表显示 | 是 |
| 3 | 考试安排查询 | 查询指定学期 | 指定学期 | 获取考试列表 | 课程名称 | 是 |
| 4 | 成绩查询 | 查询指定学期的成绩  GPA 计算 | 指定学期 | 获取成绩列表 | 成绩列表  （课程名称、平时成绩、期末成  绩、总成绩）  GPA 报告 | 是 |
| 5 | 评教 | 对指定学期轮次的教师教学评教 | 指定学期轮次  指定教学项目  评教内容 | 反馈评教信息 | 评教 | 是 |
| 6 | 自习室查询 | 查询指定时段的空闲自习室 | 指定字段 | 获取自习室列表 | 空闲自习室列表 | 是 |
| 7 | 图书馆馆藏查询 | 查询图书馆馆藏 | 图书名称 | 获取图书列表 | 相关图书列表 | 是 |
| 8 | 图书馆选座 | 查询并提交图书馆选座 | 指定座位  指定时段 | 获取选座信息 | 选座信息 | 是 |
| 9 | 北师大网关 | 登录北师大网关，连接互联网 | 用户名密码 | 获取反馈网络信息 | 登录信息流量信息 | 是 |

## 接口设计检查列表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 接口名称 | 功能描述 | 接口标准 | 入口参数 | 出口参数 | 传输频率 | 是否实现 |
| 1 | 数字京师统一身份认证平台接口 | 京师统一身份认证系统登录 | Oauth | 用户名密码 | Key | 每次开启 app 时 | 是 |
| 2 | 查看课表接口 | 显示本周或指定周课程表 | HTTP | 指定 | 课程名称  时间地点  任课教师 | 查看或导入课表时 | 是 |
| 3 | 考试安排查询接口 | 查询指定学期的考试安排 | HTTP | 指定学期 | 课程名称  时间地点  座位号 | 查询考试安排时 | 是 |
| 4 | 成绩查询接口 | 查询指定学期的成绩 | HTTP | 指定学期 | 课程名称  平时成绩  期末成绩  总成绩 | 查询成绩时 | 是 |
| 5 | 评教接口 | 对指定学期轮次的教室教学评价 | HTTP | 指定学期、教学项  目、评教内容 | 评教 | 评教时 | 是 |
| 6 | 自习室查询接口 | 查询指定时段的空闲自习室 | HTTP | 指定时段 | 空闲自习室列表 | 查询空闲自习室时 | 是 |
| 7 | 图书馆馆藏查询 | 查询图书馆馆藏 | HTTP | 图书名称 | 相关图书列表 | 查询图书馆馆藏时 | 是 |
| 8 | 图书馆选座 | 查询并提交图书馆选座 | HTTP | 指定座位  指定时段 | 选座信息 | 图书馆选座时 | 是 |
| 9 | 北师大网关 | 登陆北师大网关 | HTTP | 用户名密码 | 登录信息  流量信息 | 登陆网关时 | 是 |