GGSIPU Result Viewer - System Architecture Flowchart

Start
|
Open Website/App
|
User Loads Page
|
Frontend Requests CAPTCHA
|
Backend Connects to GGSIPU Portal
|
Establish Session with Portal
|
Fetch CAPTCHA Image from Portal
|
Store Session Cookies
|
Return CAPTCHA to Frontend
|
Display CAPTCHA to User
|
User Enters Credentials:
- Enrollment Number
- Password
- CAPTCHA Text
|
User Clicks "Login & Fetch Results"
|
Frontend Sends Login Request
|
Backend Receives Request
|
Validate Input Fields
|
Submit Credentials to GGSIPU Portal
|
GGSIPU Portal Validates CAPTCHA
|
GGSIPU Portal Validates Credentials
|
Is Valid?
|- No -> Error Message -> Refresh CAPTCHA -> Retry Login
+- Yes
|
Portal Creates Session
|
Backend Fetches Result Page HTML

|
Parse HTML with Cheerio
|
Extract Student Data:
- Name
- Enrollment No
- Programme
|
Extract Semester Results:
- Subject Codes
- Subject Names
- Internal Marks
- External Marks
- Total Marks
|
Map Subject Codes to Credits (from CSV)
|
Calculate Grade Points (based on marks)
|
Calculate SGPA (per semester)
|
Calculate Overall CGPA
|
Return Result Data to Frontend
|
Frontend Displays Results:
- Student Information
- CGPA with Progress Bar
- Semester-wise SGPA Charts
- Subject-wise Performance Graphs
- Detailed Marks Tables
|
End

```
### CAPTCHA Fetching Flow (Simplified)
```

Start
|
Generate/Use Session ID
|
Backend: Connect to GGSIPU Portal
|
Request: GET https://examweb.ggsipu.ac.in/web/login.jsp
|
Store Session Cookies from Response
|
Request: GET https://examweb.ggsipu.ac.in/web/CaptchaServlet
|

Receive CAPTCHA Image (JPEG/PNG)
|
Convert Image to Base64 Format
|
Return JSON to Frontend:
- sessionId
- captcha (base64 image)
|
Frontend: Display CAPTCHA Image
|
End

Result Fetching Flow (Simplified)

Start
|
Backend: Receive Login Credentials
|
Use Stored Session Cookies
|
POST to https://examweb.ggsipu.ac.in/web/studentlogin.do
- enrollmentNo
- password
- captcha
|
Check Response for Errors
|
Is Authentication Successful?
|- No -> Return Error Message
+- Yes
|
Try Multiple Result URLs:
- view-result.do
- viewResult.do
- result.do
- viewstudentresult.do
|
Found Valid Result Page?
|- No -> Return "Could Not Fetch Results"
+- Yes
|
Parse HTML Tables with Cheerio
|
Extract Student Information
|
Extract All Semester Data
|
For Each Semester:

- Extract Subject Details
- Map Codes to Credits (CSV lookup)
- Calculate Grade Points
- Calculate SGPA
|
Calculate Overall CGPA
|
Return Complete Result Data
|
End

Grade Calculation Flow (Simplified)

Start (For Each Subject)
|
Get Total Marks (Internal + External)
|
Determine Grade Point:
- 90-100 -> 10
- 75-89 -> 9
- 65-74 -> 8
- 55-64 -> 7
- 50-54 -> 6
- 45-49 -> 5
- 40-44 -> 4
- <40   -> 0 (Fail)
|
Lookup Credits from CSV Database
(13,200+ subject codes)
|
Credits Found?
|- No -> Use Default 3 Credits (with warning)
+- Yes -> Use Mapped Credits
|
Calculate: Grade Points × Credits
|
Add to Semester Totals
|
More Subjects?
|- Yes -> (Loop back to next subject)
+- No
|
Calculate SGPA:
= Total Grade Points ÷ Total Credits
|
More Semesters?
|- Yes -> (Process next semester)
+- No

|
Calculate CGPA:
= All Grade Points ÷ All Credits
|
End

```
---

## Overall System Architecture
```

Application Initialization Flow

```
flowchart TD
    Start([Application Start]) --> LoadPage[Load index.html]
    LoadPage --> DOMReady{DOM Content<br/>Loaded?}
    DOMReady -->|No| DOMReady
    DOMReady -->|Yes| InitApp[Initialize Application]

    InitApp --> SetupListeners[Setup Event Listeners]
    SetupListeners --> LoadCapt[Load CAPTCHA Image]

    LoadCapt --> ServerInit[Server Initialization]
    ServerInit --> LoadMiddleware[Load Middleware<br/>- CORS<br/>- JSON Parser<br/>- Cooki

    LoadMiddleware --> LoadCSV[Load Credits CSV File]
    LoadCSV --> ParseCSV[Parse 13,200+ Subject Codes<br/>and Credits]
    ParseCSV --> CreateMap[Create Credits Map<br/>creditsMap Object]

    CreateMap --> InitSessions[Initialize Session Store<br/>In-Memory Map]
    InitSessions --> RegisterRoutes[Register API Routes<br/>- /api/captcha<br/>- /api/logi

    RegisterRoutes --> StartServer[Start Express Server<br/>on Port 3000]
    StartServer --> Ready([Application Ready])

    style Start fill:#90EE90
    style Ready fill:#90EE90
    style ServerInit fill:#FFE4B5
    style LoadCSV fill:#FFE4B5
```

CAPTCHA Fetching Workflow

```
flowchart TD
    Start([User Loads Page]) --> CheckSession{Session ID<br/>Exists?}
    CheckSession -->|No| GenSession[Generate New Session ID<br/>sessionId = Date.now]
    CheckSession -->|Yes| UseSession[Use Existing Session ID]

    GenSession --> ShowLoading[Show Loading Overlay]
    UseSession --> ShowLoading

    ShowLoading --> CallAPI[Call GET /api/captcha<br/>with sessionId]

    CallAPI --> ServerReceive[Server Receives Request]
```

```
    ServerReceive --> CreateAxios[Create Axios Instance<br/>with Session Cookies]

    CreateAxios --> FetchLogin[HTTP GET to GGSIPU<br/>login.jsp]
    FetchLogin --> CheckLogin{Login Page<br/>Status 200?}

    CheckLogin -->|No| ErrorPortal[Return Error:<br/>PORTAL_UNAVAILABLE]
    CheckLogin -->|Yes| StoreCookies1[Store Session Cookies<br/>from Set-Cookie Headers]

    StoreCookies1 --> FetchCaptcha[HTTP GET to GGSIPU<br/>CaptchaServlet]
    FetchCaptcha --> CheckCaptcha{CAPTCHA Status<br/>200?}

    CheckCaptcha -->|No| ErrorCaptcha[Return Error:<br/>CAPTCHA_FETCH_FAILED]
    CheckCaptcha -->|Yes| ValidateData{Image Data<br/>Valid?}

    ValidateData -->|No| ErrorEmpty[Return Error:<br/>CAPTCHA_EMPTY]
    ValidateData -->|Yes| StoreCookies2[Store More Cookies]

    StoreCookies2 --> ToBase64[Convert Image to Base64]
    ToBase64 --> DetectMIME[Detect MIME Type<br/>from Content-Type Header]
    DetectMIME --> ReturnSuccess[Return JSON:<br/>- success: true<br/>- sessionId<br/>- ca

    ReturnSuccess --> ClientReceive[Client Receives Response]
    ClientReceive --> UpdateImage[Update CAPTCHA Image<br/>in UI]
    UpdateImage --> HideLoading[Hide Loading Overlay]
    HideLoading --> End([CAPTCHA Displayed])

    ErrorPortal --> ShowPlaceholder[Show CAPTCHA Placeholder<br/>with Refresh Button]
    ErrorCaptcha --> ShowPlaceholder
    ErrorEmpty --> ShowPlaceholder
    ShowPlaceholder --> ErrorEnd([Error State])

    style Start fill:#90EE90
    style End fill:#90EE90
    style ErrorEnd fill:#FFB6C1
    style FetchLogin fill:#FFE4B5
    style FetchCaptcha fill:#FFE4B5
```

Login and Authentication Flow

```
flowchart TD
    Start([User Submits Form]) --> ValidateInput{All Fields<br/>Filled?}
    ValidateInput -->|No| ShowError1[Show Error:<br/>Fill All Fields]
    ValidateInput -->|Yes| ShowLoading[Show Loading Overlay]

    ShowLoading --> PrepareData[Prepare Login Data:<br/>- enrollmentNo<br/>- password<br/>

    PrepareData --> CallAPI[POST /api/login]
    CallAPI --> ServerReceive[Server Receives Request]

    ServerReceive --> ValidateServer{Required Fields<br/>Present?}
    ValidateServer -->|No| Error400[Return 400:<br/>Missing Required Fields]

    ValidateServer -->|Yes| GetAxios[Get Axios Instance<br/>with Session Cookies]
    GetAxios --> PrepareForm[Prepare Form Data<br/>Multiple Field Name Variants:<br/>- enr

    PrepareForm --> PostLogin[HTTP POST to GGSIPU<br/>studentlogin.do]
    PostLogin --> StoreC1[Store Response Cookies]
    StoreC1 --> CheckResponse{Response Contains<br/>Invalid/Incorrect/Wrong?}
```

```
    CheckResponse -->|Yes| ReturnFail[Return JSON:<br/>success: false<br/>Invalid Credenti
    CheckResponse -->|No| TryResults[Try Multiple Result URLs:<br/>- view-result.do<br/>-

    TryResults --> LoopURLs[Loop Through URLs]
    LoopURLs --> FetchResult[HTTP GET Result URL]
    FetchResult --> StoreC2[Store Cookies]
    StoreC2 --> CheckResult{Status 200 &<br/>Contains table?}

    CheckResult -->|No| NextURL{More URLs<br/>to Try?}
    NextURL -->|Yes| LoopURLs
    NextURL -->|No| ReturnNoResult[Return Error:<br/>Could Not Fetch Results]

    CheckResult -->|Yes| ParseHTML[Parse Result HTML<br/>with Cheerio]
    ParseHTML --> CalcGrades[Calculate CGPA/SGPA]
    CalcGrades --> ReturnSuccess[Return JSON:<br/>success: true<br/>data: resultData]

    ReturnSuccess --> ClientReceive[Client Receives Response]
    ClientReceive --> CheckSuccess{Login<br/>Successful?}

    CheckSuccess -->|No| ShowError2[Show Error Message<br/>Refresh CAPTCHA]
    CheckSuccess -->|Yes| StoreResult[Store Result Data<br/>in resultData Variable]
    StoreResult --> DisplayResults[Call displayResults]
    DisplayResults --> HideLogin[Hide Login Section]
    HideLogin --> ShowResults[Show Results Section]
    ShowResults --> PopulateInfo[Populate Student Info]
    PopulateInfo --> UpdateCGPA[Update CGPA Display]
    UpdateCGPA --> DrawCharts[Draw Charts]
    DrawCharts --> ShowSemesters[Show Semester Tables]
    ShowSemesters --> ScrollTop[Scroll to Top]
    ScrollTop --> HideLoading[Hide Loading Overlay]
    HideLoading --> End([Results Displayed])

    ShowError1 --> ErrorEnd([Error State])
    ShowError2 --> ErrorEnd
    ReturnFail --> ErrorEnd
    ReturnNoResult --> ErrorEnd
    Error400 --> ErrorEnd

    style Start fill:#90EE90
    style End fill:#90EE90
    style ErrorEnd fill:#FFB6C1
    style PostLogin fill:#FFE4B5
    style FetchResult fill:#FFE4B5
```

Result Fetching and Parsing Flow

```
flowchart TD
    Start([HTML Data Received]) --> LoadCheerio[Load HTML with Cheerio]
    LoadCheerio --> InitResult[Initialize Result Object:<br/>- studentName<br/>- enrollmen

    InitResult --> ExtractInfo[Extract Student Information]

    subgraph "Student Info Extraction"
        ExtractInfo --> ScanCells[Scan All td, th, span, div]
        ScanCells --> FindName{Text Matches<br/>'Student Name'?}
        FindName -->|Yes| GetName[Get Next Cell Value<br/>as Student Name]
        FindName -->|No| FindEnroll{Text Matches<br/>'Enrollment No'?}
```

```
    FindEnroll -->|Yes| GetEnroll[Get Next Cell Value<br/>as Enrollment No]
    FindEnroll -->|No| FindProg{Text Matches<br/>'Programme'?}
    FindProg -->|Yes| GetProg[Get Next Cell Value<br/>as Programme]
    FindProg -->|No| Continue[Continue Scanning]
end

Continue --> ParseTables[Find All table Elements]
ParseTables --> InitSem[Initialize Semester Counter = 1]

InitSem --> LoopTables[Loop Through Each Table]
LoopTables --> InitSubjects[Initialize Empty Subjects Array]
InitSubjects --> FindHeaders[Scan First Row for Headers]

subgraph "Identify Columns"
    FindHeaders --> CheckCode{Header Contains<br/>'code' or 'sub no'?}
    CheckCode -->|Yes| SaveCodeCol[Save Code Column Index]
    CheckCode -->|No| CheckName{Header Contains<br/>'subject name'?}
    CheckName -->|Yes| SaveNameCol[Save Name Column Index]
    CheckName -->|No| CheckInt{Header Contains<br/>'internal' or 'mid'?}
    CheckInt -->|Yes| SaveIntCol[Save Internal Column Index]
    CheckInt -->|No| CheckExt{Header Contains<br/>'external' or 'end'?}
    CheckExt -->|Yes| SaveExtCol[Save External Column Index]
    CheckExt -->|No| CheckTotal{Header Contains<br/>'total' or 'grand'?}
    CheckTotal -->|Yes| SaveTotalCol[Save Total Column Index]
    CheckTotal -->|No| NextHeader[Next Header]
end

NextHeader --> ParseRows[Parse Data Rows]
ParseRows --> LoopRows[Loop Through Rows]
LoopRows --> GetCells[Get All Cells in Row]

GetCells --> ExtractValues[Extract Values:<br/>- Subject Code<br/>- Subject Name<br/>-

ExtractValues --> ValidateCode{Subject Code<br/>Matches Pattern?}
ValidateCode -->|No| SkipRow[Skip Row]
ValidateCode -->|Yes| ValidateTotal{Total > 0?}
ValidateTotal -->|No| CalcTotal[Calculate Total =<br/>Internal + External]
ValidateTotal -->|Yes| AddSubject[Add Subject to Array]
CalcTotal --> AddSubject

AddSubject --> MoreRows{More Rows<br/>in Table?}
MoreRows -->|Yes| LoopRows
MoreRows -->|No| CheckSubjects{Found Any<br/>Subjects?}

CheckSubjects -->|Yes| CreateSemester[Create Semester Object:<br/>- semester: number<b
CheckSubjects -->|No| SkipTable[Skip This Table]

CreateSemester --> AddSemester[Add to Semesters Array]
AddSemester --> IncrementSem[Increment Semester Counter]
IncrementSem --> MoreTables{More Tables?}

SkipRow --> MoreRows
SkipTable --> MoreTables

MoreTables -->|Yes| LoopTables
MoreTables -->|No| ReturnResult[Return Parsed Result Object]
ReturnResult --> End([Parsing Complete])

style Start fill:#90EE90
style End fill:#90EE90
style LoadCheerio fill:#FFE4B5
```

CGPA/SGPA Calculation Logic

```
flowchart TD
    Start([Result Data Received]) --> InitTotals[Initialize:<br/>totalCredits = 0<br/>tota

    InitTotals --> LoopSemesters[Loop Through Each Semester]
    LoopSemesters --> InitSemester[Initialize for Semester:<br/>semesterCredits = 0<br/>se

    InitSemester --> LoopSubjects[Loop Through Each Subject]
    LoopSubjects --> GetMarks[Get Subject Total Marks]

    GetMarks --> CalcGrade[Calculate Grade Point]

    subgraph "Grade Point Calculation"
        CalcGrade --> Check90{Marks >= 90?}
        Check90 -->|Yes| Grade10[Grade Point = 10]
        Check90 -->|No| Check75{Marks >= 75?}
        Check75 -->|Yes| Grade9[Grade Point = 9]
        Check75 -->|No| Check65{Marks >= 65?}
        Check65 -->|Yes| Grade8[Grade Point = 8]
        Check65 -->|No| Check55{Marks >= 55?}
        Check55 -->|Yes| Grade7[Grade Point = 7]
        Check55 -->|No| Check50{Marks >= 50?}
        Check50 -->|Yes| Grade6[Grade Point = 6]
        Check50 -->|No| Check45{Marks >= 45?}
        Check45 -->|Yes| Grade5[Grade Point = 5]
        Check45 -->|No| Check40{Marks >= 40?}
        Check40 -->|Yes| Grade4[Grade Point = 4]
        Check40 -->|No| Grade0[Grade Point = 0 - FAIL]
    end

    Grade10 --> LookupCredits[Lookup Subject Code in CSV]
    Grade9 --> LookupCredits
    Grade8 --> LookupCredits
    Grade7 --> LookupCredits
    Grade6 --> LookupCredits
    Grade5 --> LookupCredits
    Grade4 --> LookupCredits
    Grade0 --> LookupCredits

    LookupCredits --> CheckFound{Credits Found<br/>in Map?}
    CheckFound -->|No| WarnDefault[Log Warning<br/>Use Default 3 Credits]
    CheckFound -->|Yes| UseCredits[Use Mapped Credits]

    WarnDefault --> SetCredits[Set subject.credits]
    UseCredits --> SetCredits

    SetCredits --> SetGrade[Set subject.gradePoint]
    SetGrade --> CalcSemGP[semesterGradePoints +=<br/>gradePoint × credits]
    CalcSemGP --> AddSemCredits[semesterCredits += credits]

    AddSemCredits --> MoreSubjects{More Subjects<br/>in Semester?}
    MoreSubjects -->|Yes| LoopSubjects
    MoreSubjects -->|No| CalcSGPA[Calculate SGPA:<br/>semesterGradePoints ÷ semesterCredit

    CalcSGPA --> RoundSGPA[Round to 2 Decimal Places]
    RoundSGPA --> SetSGPA[Set semester.sgpa]
    SetSGPA --> SetSemCredits[Set semester.totalCredits]
```

```
    SetSemCredits --> AddTotals[totalCredits += semesterCredits<br/>totalGradePoints += se

    AddTotals --> MoreSemesters{More Semesters?}
    MoreSemesters -->|Yes| LoopSemesters
    MoreSemesters -->|No| CalcCGPA[Calculate CGPA:<br/>totalGradePoints ÷ totalCredits]

    CalcCGPA --> RoundCGPA[Round to 2 Decimal Places]
    RoundCGPA --> SetCGPA[Set resultData.cgpa]
    SetCGPA --> SetTotalCredits[Set resultData.totalCredits]
    SetTotalCredits --> End([Calculation Complete])

    style Start fill:#90EE90
    style End fill:#90EE90
    style Grade0 fill:#FFB6C1
    style WarnDefault fill:#FFFFE0
```

Data Visualization Flow

```
flowchart TD
    Start([Display Results Called]) --> ShowResults[Show Results Section]
    ShowResults --> PopulateInfo[Populate Student Information:<br/>- Name<br/>- Enrollment

    PopulateInfo --> DisplayCGPA[Display CGPA Value]
    DisplayCGPA --> CalcPercentage[Calculate CGPA Percentage<br/>= cgpa / 10 × 100]
    CalcPercentage --> UpdateBar[Update CGPA Progress Bar Width]
    UpdateBar --> ShowCredits[Display Total Credits]

    ShowCredits --> CreateSGPAChart[Create SGPA Bar Chart]

    subgraph "SGPA Chart Creation"
        CreateSGPAChart --> GetCtx1[Get Canvas Context<br/>sgpaChart]
        GetCtx1 --> PrepareLabels[Prepare Labels:<br/>Sem 1, Sem 2, ...]
        PrepareLabels --> PrepareData1[Prepare Data:<br/>SGPA Values Array]
        PrepareData1 --> DestroyOld1{Existing Chart?}
        DestroyOld1 -->|Yes| Destroy1[Destroy Old Chart]
        DestroyOld1 -->|No| CreateNew1[Create New Chart.js Instance]
        Destroy1 --> CreateNew1
        CreateNew1 --> ConfigBar[Configure Bar Chart:<br/>- Type: bar<br/>- Color: Blue Gr
    end

    ConfigBar --> CreateSubjectChart[Create Subject Performance Chart]

    subgraph "Subject Chart Creation"
        CreateSubjectChart --> GetCtx2[Get Canvas Context<br/>subjectChart]
        GetCtx2 --> CollectSubjects[Collect All Subjects<br/>from All Semesters]
        CollectSubjects --> LimitSubjects[Limit to First 10 Subjects<br/>for Better Visual
        LimitSubjects --> PrepareLabels2[Prepare Labels:<br/>Subject Codes]
        PrepareLabels2 --> PrepareData2[Prepare Data:<br/>Grade Points Array]
        PrepareData2 --> DestroyOld2{Existing Chart?}
        DestroyOld2 -->|Yes| Destroy2[Destroy Old Chart]
        DestroyOld2 -->|No| CreateNew2[Create New Chart.js Instance]
        Destroy2 --> CreateNew2
        CreateNew2 --> ConfigLine[Configure Line Chart:<br/>- Type: line<br/>- Color: Purp
    end

    ConfigLine --> CreateTables[Create Semester Tables]

    subgraph "Table Creation"
```

```
        CreateTables --> ClearContainer[Clear semesterResults Container]
        ClearContainer --> LoopSemesters[Loop Through Each Semester]
        LoopSemesters --> CreateCard[Create Semester Card]
        CreateCard --> AddHeader[Add Semester Header:<br/>- Semester Number<br/>- SGPA Bad
        AddHeader --> CreateTable[Create HTML Table]
        CreateTable --> AddTableHeader[Add Table Headers:<br/>- Subject Code<br/>- Subject
        AddTableHeader --> LoopSubjects[Loop Through Subjects]
        LoopSubjects --> CreateRow[Create Table Row]
        CreateRow --> AddCells[Add Cells with Data]
        AddCells --> StyleGrade[Apply Grade Color:<br/>- O: Green<br/>- A: Light Green<br/
        StyleGrade --> MoreSubjects{More Subjects?}
        MoreSubjects -->|Yes| LoopSubjects
        MoreSubjects -->|No| AppendTable[Append Table to Card]
        AppendTable --> AppendCard[Append Card to Container]
        AppendCard --> MoreSemesters{More Semesters?}
        MoreSemesters -->|Yes| LoopSemesters
    end

    MoreSemesters -->|No| ScrollTop[Scroll Page to Top<br/>Smooth Scroll]
    ScrollTop --> End([Visualization Complete])

    style Start fill:#90EE90
    style End fill:#90EE90
    style CreateNew1 fill:#E6F3FF
    style CreateNew2 fill:#E6F3FF
```

Demo Mode Flow

```
flowchart TD
    Start([User Clicks Demo Button]) --> ShowLoading[Show Loading Overlay]
    ShowLoading --> CallAPI[Call GET /api/demo]

    CallAPI --> ServerReceive[Server Receives Request]
    ServerReceive --> CreateMockData[Create Mock Result Data]

    subgraph "Mock Data Structure"
        CreateMockData --> SetStudent[Set Student Info:<br/>- Name: VIJAY KUMAR<br/>- Enro
        SetStudent --> CreateSem1[Create Semester 1:<br/>5 Subjects with Marks]
        CreateSem1 --> CreateSem2[Create Semester 2:<br/>5 Subjects with Marks]
        CreateSem2 --> CreateSem3[Create Semester 3:<br/>5 Subjects with Marks]
        CreateSem3 --> CreateSem4[Create Semester 4:<br/>5 Subjects with Marks]
        CreateSem4 --> AssignCodes[Assign Subject Codes:<br/>ES-101, ES-102, etc.]
        AssignCodes --> AssignMarks[Assign Internal & External Marks<br/>Total = 70-96 ran
    end

    AssignMarks --> CalcDemo[Calculate CGPA/SGPA<br/>Using Same Logic]
    CalcDemo --> ReturnJSON[Return JSON:<br/>success: true<br/>data: demoData]

    ReturnJSON --> ClientReceive[Client Receives Response]
    ClientReceive --> CheckSuccess{Success?}

    CheckSuccess -->|No| ShowError[Show Error Message]
    CheckSuccess -->|Yes| StoreData[Store Result Data]
    StoreData --> DisplayResults[Call displayResults Function]

    DisplayResults --> SameFlow[Follow Same Display Flow<br/>as Real Login]
    SameFlow --> HideLoading[Hide Loading Overlay]
    HideLoading --> End([Demo Results Displayed])
```

```
    ShowError --> ErrorEnd([Error State])

    style Start fill:#90EE90
    style End fill:#90EE90
    style ErrorEnd fill:#FFB6C1
    style CreateMockData fill:#E6FFE6
```

Error Handling Flow

```
flowchart TD
    Start([Error Occurs]) --> IdentifyType{Error Type}

    IdentifyType -->|Network Error| NetworkError[Error Code:<br/>- ENOTFOUND<br/>- ECONNRE
    IdentifyType -->|Portal Error| PortalError[Portal Status:<br/>- 500+ Status<br/>- Port
    IdentifyType -->|Client Error| ClientError[Client Validation:<br/>- Missing Fields<br/
    IdentifyType -->|Authentication Error| AuthError[Login Failed:<br/>- Invalid Credentia
    IdentifyType -->|Parsing Error| ParseError[HTML Parsing:<br/>- Structure Changed<br/>-

    NetworkError --> CheckCode{Specific<br/>Error Code?}
    CheckCode -->|ENOTFOUND| Msg1[Message: Unable to Connect<br/>Check Internet Connection
    CheckCode -->|ETIMEDOUT| Msg2[Message: Connection Timed Out<br/>Try Again]
    CheckCode -->|Other| Msg3[Message: Network Error<br/>Please Try Again]

    PortalError --> CheckStatus{Status Code?}
    CheckStatus -->|503| Msg4[Message: Portal Unavailable<br/>Use Demo Mode]
    CheckStatus -->|500-599| Msg5[Message: Portal Experiencing Issues<br/>Try Later]
    CheckStatus -->|403| Msg6[Message: Access Forbidden<br/>Rate Limited]

    ClientError --> Msg7[Message: Please Fill All Fields<br/>Check Form]

    AuthError --> CheckType{Auth Error<br/>Type?}
    CheckType -->|Invalid Login| Msg8[Message: Invalid Credentials<br/>Try Again]
    CheckType -->|Wrong CAPTCHA| Msg9[Message: Incorrect CAPTCHA<br/>Refresh & Retry]
    CheckType -->|Session Expired| Msg10[Message: Session Expired<br/>Reload Page]

    ParseError --> CheckParse{Parsing<br/>Issue?}
    CheckParse -->|No Tables| Msg11[Message: No Results Found<br/>Check Enrollment No]
    CheckParse -->|No Subjects| Msg12[Message: Could Not Parse Results<br/>Portal Structur

    Msg1 --> LogError[Log Error to Console]
    Msg2 --> LogError
    Msg3 --> LogError
    Msg4 --> LogError
    Msg5 --> LogError
    Msg6 --> LogError
    Msg7 --> LogError
    Msg8 --> LogError
    Msg9 --> LogError
    Msg10 --> LogError
    Msg11 --> LogError
    Msg12 --> LogError

    LogError --> ShowUI[Show Error in UI:<br/>- Red Error Box<br/>- Error Message<br/>- Su

    ShowUI --> OfferAction{Can User<br/>Retry?}
    OfferAction -->|Yes| EnableRetry[Enable Retry Actions:<br/>- Refresh CAPTCHA<br/>- Try
    OfferAction -->|No| SuggestHelp[Suggest:<br/>- Contact Support<br/>- Check Portal Stat
```

```
    EnableRetry --> HideLoading[Hide Loading Overlay]
    SuggestHelp --> HideLoading
    HideLoading --> End([Error Handled])

    style Start fill:#FFB6C1
    style End fill:#90EE90
    style LogError fill:#FFFFE0
```

Technology Stack

Frontend

- **HTML5**: Structure and layout
- **CSS3**: Styling with gradients, animations, responsive design
- **JavaScript (ES6+)**: Client-side logic, event handling, API calls
- **Chart.js 4.4.0**: Data visualization (bar charts, line charts)

Backend

- **Node.js**: JavaScript runtime
- **Express.js 4.18.2**: Web framework
- **Axios 1.6.2**: HTTP client for GGSIPU portal requests
- **Cheerio 1.1.2**: Server-side HTML parsing (jQuery-like syntax)
- **Cookie-Parser 1.4.6**: Parse and manage session cookies
- **CORS 2.8.5**: Enable cross-origin requests

Data

- **CSV File**: 13,200+ subject codes mapped to credits
- **In-Memory Map**: Session storage (sessions Map object)

External Integration

- **GGSIPU Portal**: examweb.ggsipu.ac.in
- Login page: /web/login.jsp
- CAPTCHA: /web/CaptchaServlet
- Authentication: /web/studentlogin.do
- Results: Multiple possible URLs

Data Flow Summary

Request Flow

```
User Browser -> Frontend (app.js) -> Express Server (server.js) -> GGSIPU Portal
```

Response Flow

```
GGSIPU Portal -> Express Server -> Parse with Cheerio -> Calculate Grades -> Frontend -> C
```

Session Management

```
Client Session ID <-> Server Session Store <-> GGSIPU Portal Cookies
```

Key Features Documented

1. **Real-time CAPTCHA**: Fetches actual CAPTCHA from GGSIPU portal
2. **Session Persistence**: Maintains cookies across requests
3. **Intelligent Parsing**: Handles various HTML table structures
4. **Credit Mapping**: Maps 13,200+ subjects to official credits
5. **Auto Calculation**: Computes SGPA and CGPA automatically
6. **Responsive Visualizations**: Interactive charts with Chart.js
7. **Error Recovery**: Comprehensive error handling with retry mechanisms
8. **Demo Mode**: Test interface without portal access
9. **Multiple URL Attempts**: Tries various result page URLs
10. **Fallback Credits**: Uses default 3 credits when code not found

---

Security Considerations

1. **HTTPS Agent**: Configurable SSL certificate validation
2. **No Credential Storage**: Credentials not stored on server
3. **Session Isolation**: Each user gets unique session
4. **Cookie Management**: Proper cookie handling and isolation
5. **Input Validation**: Both client and server-side validation
6. **Error Sanitization**: No sensitive data in error messages
7. **CORS Protection**: Configured cross-origin policies
8. **Timeout Handling**: 30-second request timeout

Deployment Architecture

```
graph LR
    A[User] --> B[Load Balancer]
    B --> C[Web Server Instance 1]
    B --> D[Web Server Instance 2]
    B --> E[Web Server Instance N]

    C --> F[GGSIPU Portal]
    D --> F
    E --> F

    C --> G[Shared Session Store<br/>Redis/Database]
    D --> G
    E --> G

    style A fill:#E1F5FF
    style F fill:#FFE1E1
    style G fill:#E6FFE6
```

Note: Current implementation uses in-memory sessions. For production with multiple instances, implement Redis

or database-backed session storage.

Future Enhancements Identified

1. **Persistent Sessions**: Redis or database-backed storage
2. **Caching**: Cache subject credits in memory/Redis
3. **Rate Limiting**: Implement request rate limiting
4. **PDF Export**: Generate PDF reports of results
5. **Comparison**: Compare performance across semesters
6. **Predictions**: Predict future CGPA based on trends
7. **Mobile App**: Native mobile application
8. **Notifications**: Email/SMS notifications for new results
9. **Analytics**: Track usage patterns and errors
10. **Multi-University**: Support other university portals
