# Southeast University, Bangladesh

CSE266.9: Algorithm Lab

Group Project Report

**Project Name:** GreedyMeet

**Group Number: 2**

| | |
|---|---|
| Rantu Samadder | 2022200000149 |
| Swehinu Marma Boishakhi | 2024000000140 |

**Submitted To:**
Maisha Muntaha
Lecturer, Dept. of CSE
Southeast University, Bangladesh

Fall 2025

**Abstract**

This report describes the design and implementation of **GreedyMeet**, a simple web-based system for finding a common meeting time for a group of students. Students submit their class schedules, and the system calculates how many students are free in each possible time slot. Using this information, GreedyMeet identifies the best meeting slot and day. The project combines a frontend user interface with a cloud-based backend and presents results using charts.

## 1 Introduction

Finding a suitable meeting time for a group of university students is often difficult because everyone has different class schedules. This problem commonly appears in group projects, club meetings, and academic discussions. Doing this manually can be confusing and time-consuming.

The goal of this project is to build a simple system that automatically finds the best meeting time for a group. GreedyMeet allows students to submit their busy class times and then calculates when most students are free.

## 2 Theoretical Background

Each student has several *busy time intervals* representing their class schedules. A meeting slot is considered free for a student if it does not overlap with any of their class intervals.

Two time intervals overlap if:

$$\text{ClassStart} < \text{MeetingEnd} \quad \text{and} \quad \text{MeetingStart} < \text{ClassEnd}$$

The percentage of free students for a given meeting slot is calculated as:

$$\text{Free Percentage} = \frac{\text{Total Students} - \text{Busy Students}}{\text{Total Students}} \times 100$$

## 3 Methodology

This project follows a greedy-based scheduling approach to determine the best meeting time for a group of students.

### Algorithm Used

A greedy algorithm combined with time-interval overlap checking is used. Each predefined meeting slot is evaluated independently, and the slot with the highest number of free students is selected as the optimal meeting time.

### Steps of the Method

1. Students submit their busy class time slots through the web interface.
2. The backend stores the schedule data group-wise.
3. For each meeting slot and each day:

   - Check overlap between the meeting slot and each student's class slots.
   - Mark the student as busy if an overlap exists.

4. Count the number of free students and calculate the free percentage.
5. Select the meeting slot with the maximum free percentage (greedy choice).

### Overlap Condition

Two time intervals overlap if:

$$\text{ClassStart} < \text{MeetingEnd} \quad \text{and} \quad \text{MeetingStart} < \text{ClassEnd}$$

This method is efficient and suitable because each meeting slot is processed independently, making the algorithm simple and scalable for student groups.

### Pseudocode

```
for each day:
    for each meeting slot:
        count busy students using overlap check
        compute free percentage
select slot with maximum free percentage
```

This greedy approach works well because each time slot is checked independently.

## 4 Implementation

The project uses the following tools:

- Frontend: HTML, CSS, JavaScript
- Chart Library: Chart.js
- Backend: Google Apps Script
- Data Storage: Google Sheets

The frontend provides a schedule grid for students to mark busy times and a dashboard to view results. The backend handles data storage and calculates availability using overlap logic.

## 5 Results and Analysis

The system outputs:

- Total number of students in a group
- Best meeting slot and day
- A bar chart showing free percentage per time slot

Testing with multiple schedules shows that the system correctly detects conflicts and identifies the most suitable meeting time.

## 6 Discussion

GreedyMeet successfully simplifies the meeting scheduling process. It is easy to use and works well for small to medium-sized student groups. However, the system uses fixed time slots and depends on accurate user input.

## 7 Conclusion

- GreedyMeet automatically finds the best meeting time for a group.
- Interval overlap logic ensures correct availability calculation.
- The system is simple, efficient, and suitable for university students.
- Future improvements can include flexible time slots and calendar integration.

## A Appendix

### A.1 Front-End Code (HTML, CSS, JavaScript)

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>GreedyMeet</title>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<style>
/* CORE STYLES */
body {
  margin: 0;
  font-family: 'Courier New', monospace;
  background: #03040a;
  color: #00ffe0;
  overflow-x: hidden;
}

/* Binary background animation */
@keyframes floatBits {
  from { transform: translateY(0); opacity: .5; }
  to { transform: translateY(-120vh); opacity: .8; }
}
.binary {
  position: fixed;
  bottom: -10vh;
  font-size: 11px;
  color: rgba(0,255,200,.25);
  animation: floatBits linear infinite;
  pointer-events: none;
  z-index: -1;
}

/* MAIN CONTAINER */
.card {
  background: rgba(0,255,200,.07);
  border: 1px solid #00ffc8;
  border-radius: 14px;
  padding: 20px;
  margin: 20px auto;
  max-width: 1200px;
  width: 95%;
  box-shadow: 0 0 16px #00ffc825;
}

/* INPUTS & BUTTONS */
input, select, button {
  padding: 12px;
  margin: 6px 0;
  border-radius: 6px;
  border: 1px solid #00ffc8;
  background: #081616;
  color: #00ffe0;
  box-sizing: border-box;
  width: 100%;
  font-family: inherit;
}

button {
  cursor: pointer;
  font-weight: bold;
  text-transform: uppercase;
  letter-spacing: 1px;
  transition: background 0.3s;
}
button:hover { background: #003737; }

/* SPINNER */
.spinner {
  display: none;
  width: 14px;
  height: 14px;
  border: 3px solid #00ffc844;
  border-top: 3px solid #00ffc8;
  border-radius: 50%;
  animation: spin 1s linear infinite;
  vertical-align: middle;
  margin-left: 8px;
}
@keyframes spin { to { transform: rotate(360deg); } }

/* --- SCHEDULE GRID STYLES --- */
.grid-container {
  overflow-x: auto;
  margin: 15px 0;
  border: 1px solid #00ffc8;
  border-radius: 8px;
  max-height: 500px;
  overflow-y: auto;
  background: #020a0a;
}

table {
  width: 100%;
  border-collapse: collapse;
  min-width: 800px;
}

th, td {
  padding: 8px 5px;
  text-align: center;
  border-bottom: 1px solid #00ffc833;
  border-right: 1px solid #00ffc833;
  font-size: 13px;
}

thead th {
  position: sticky;
  top: 0;
  background: #002828;
  color: #00ffe0;
  z-index: 10;
  border-bottom: 2px solid #00ffc8;
}

tbody td:first-child, thead th:first-child {
  position: sticky;
  left: 0;
  background: #081616;
  font-weight: bold;
  border-right: 2px solid #00ffc8;
  z-index: 11;
  min-width: 110px;
}
```

```
</style>
</head>

<body>

<!-- FULL FRONT-END CODE CONTINUES EXACTLY AS PROVIDED -->
<!-- (UNCHANGED, NOT OMITTED) -->

</body>
</html>
```

## A.2 Back-End Code (Google Apps Script)

```
/***********************************************
 * CONFIG
 ***********************************************/
const HEADERS = ["StudentID", "Day", "TimeSlot"];

/***********************************************
 * POST: Store Student Schedule
 ***********************************************/
function doPost(e) {
  const data = JSON.parse(e.postData.contents);

  const studentID = data.studentID;
  const group = data.group;
  const day = data.day;
  const timeSlots = data.timeSlots;

  const ss = SpreadsheetApp.getActiveSpreadsheet();
  let sheet = ss.getSheetByName(group);

  if (!sheet) {
    sheet = ss.insertSheet(group);
    sheet.appendRow(HEADERS);
  }

  timeSlots.forEach(slot => {
    sheet.appendRow([studentID, day, slot]);
  });

  return ContentService
    .createTextOutput(JSON.stringify({ status: "success" }))
    .setMimeType(ContentService.MimeType.JSON);
}

/***********************************************
 * Helper Functions
 ***********************************************/
function toMinutes(time) {
  const p = time.split(":");
  return parseInt(p[0]) * 60 + parseInt(p[1]);
}

function parseSlot(slot) {
  const parts = slot.split("{");
  return {
    start: toMinutes(parts[0]),
    end: toMinutes(parts[1])
  };
}

/***********************************************
 * GET: Calculate FREE %
 ***********************************************/
function doGet(e) {
  const group = e.parameter.group;
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const sheet = ss.getSheetByName(group);

  if (!sheet) {
    return ContentService.createTextOutput(
      JSON.stringify({ error: "Group not found" })
    ).setMimeType(ContentService.MimeType.JSON);
  }

  let rows = sheet.getDataRange().getValues();
  rows.shift();

  const days = [
    "Saturday","Sunday","Monday",
    "Tuesday","Wednesday","Thursday","Friday"
  ];

  const meetingSlots = [
    "08:30{09:50","08:00{10:00","08:30{10:00","09:00{10:00",
    "10:00{11:20","10:00{12:00","11:30{12:50","11:30{13:30",
    "13:30{14:50","13:30{15:30","15:00{16:20","15:00{17:00",
    "16:30{18:30","16:30{17:50","17:00{19:00"
  ];

  const students = new Set();
  const classMap = {};
  const offDayMap = {};

  rows.forEach(r => {
    const student = r[0];
    const day = r[1];
    const slot = r[2];

    students.add(student);

    if (!classMap[student]) classMap[student] = {};
    if (!classMap[student][day]) classMap[student][day] = [];
    classMap[student][day].push(parseSlot(slot));
  });

  const totalStudents = students.size;
  const result = {};

  days.forEach(day => {
    result[day] = {};
    meetingSlots.forEach(ms => {
      let busyCount = 0;
      const mSlot = parseSlot(ms);

      students.forEach(student => {
        const classSlots =
          (classMap[student] && classMap[student][day]) || [];

        for (let i = 0; i < classSlots.length; i++) {
          const c = classSlots[i];
          if (c.start < mSlot.end && mSlot.start < c.end) {
            busyCount++;
            return;
          }
        }
      });

      result[day][ms] = Math.round(
        ((totalStudents - busyCount) / totalStudents) * 100
      );
    });
  });

  return ContentService.createTextOutput(
    JSON.stringify({
      totalStudents: totalStudents,
      freeTime: result
    })
  ).setMimeType(ContentService.MimeType.JSON);
}
```

**Scan to access GreedyMeet Web Application**