Daily Reporting — IST Time Fixes & Frontend Improvements

This PDF consolidates the backend (Apps Script) fixes for IST-safe date/time handling and the frontend (React) improvements to DailyReportTimetable.jsx. It includes ready-to-paste helpers, clearly marked replacement snippets (surgical edits), and optional UX upgrades.

BACKEND — Add IST-safe helpers (paste once near top)

```
/** ====== TIME / DATE HELPERS (IST SAFE) ====== */
const TZ = 'Asia/Kolkata';

/** Returns "YYYY-MM-DD" for *today* in IST */
function _todayISO() {
  const now = new Date();
  return Utilities.formatDate(now, TZ, 'yyyy-MM-dd');
}

/** Coerce any sheet cell (Date | string | number) to a JS Date (best effort) */
function _coerceToDate(value) {
  if (value instanceof Date) return value;
  if (typeof value === 'number') return new Date(value); // serial/time value
  if (typeof value === 'string' && value.trim()) {
    const d = new Date(value);
    if (!isNaN(d.getTime())) return d;
  }
  return null;
}

/** Return IST "YYYY-MM-DD" for any sheet cell (Date/string/number). Null-safe. */
function _isoDateIST(value) {
  const d = _coerceToDate(value);
  if (!d) return '';
  return Utilities.formatDate(d, TZ, 'yyyy-MM-dd');
}

/** Get day name in IST from any date-like input (e.g., "Monday") */
function _dayNameIST(value) {
  const d = _coerceToDate(value) || new Date();
  return Utilities.formatDate(d, TZ, 'EEEE');
}

/** Parse client date ("YYYY-MM-DD" or ISO) to IST "YYYY-MM-DD" */
function _normalizeQueryDate(dateStr) {
  if (!dateStr) return _todayISO();
  const d = new Date(dateStr);
  if (isNaN(d.getTime())) return _todayISO();
  return Utilities.formatDate(d, TZ, 'yyyy-MM-dd');
}
```

BACKEND — Surgical replacements (use helpers everywhere)

A) debugSubstitutionsForDate

Before:

```
const date = e.parameter.date || new Date().toISOString().split('T')[0];
const normalizedQueryDate = _isoDateString(date);
...
const matches = allRows.filter(r => {
  const rowDate = _isoDateString(r.date);
  return rowDate === normalizedQueryDate;
});
```

After:

```
const date = _normalizeQueryDate(e.parameter.date);
const normalizedQueryDate = date;
...
const matches = allRows.filter(r => _isoDateIST(r.date) === normalizedQueryDate);
```

BACKEND — B) submitDailyReport duplicate check

Before:

```
const reportDate = String(data.date || '').split('T')[0];
...
const duplicate = existingReports.find(r => {
  let rDate;
  const dateValue = r.date;
  if (dateValue instanceof Date) {
    const year = dateValue.getFullYear();
    const month = String(dateValue.getMonth() + 1).padStart(2, '0');
    const day = String(dateValue.getDate()).padStart(2, '0');
    rDate = year + '-' + month + '-' + day;
  } else {
    rDate = String(dateValue || '').split('T')[0];
  }
  return rDate === reportDate && ...;
});
```

After:

```
const reportDate = _normalizeQueryDate(data.date);
...
const duplicate = existingReports.find(r => {
  const rDate = _isoDateIST(r.date);
  return rDate === reportDate && ...;
});
```

BACKEND — C) _handleGetTeacherDailyReportsForDate

Before:

```
const date = params.date || _todayISO();
...
let reportDate;
const dateValue = report.date;
if (dateValue instanceof Date) { ... } else {
  reportDate = String(dateValue || '').split('T')[0];
}
const queryDate = String(date).split('T')[0];
return reportDate === queryDate && reportEmail === email;
```

After:

```
const date = _normalizeQueryDate(params.date);
...
const reportDate = _isoDateIST(report.date);
const queryDate = date;
return reportDate === queryDate && reportEmail === email;
```

BACKEND — D) _handleGetDailyReportsForDate

Before:

```
const date = params.date || _todayISO();
const dayName = _dayName(date);
...
const allReports = _rows(drSh)...filter(report => {
  let reportDate; /* manual build or split('T') */
  return reportDate === date;
});
```

After:

```
const date = _normalizeQueryDate(params.date);
const dayName = _dayNameIST(date);
...
const allReports = _rows(drSh)
  .map(r => _indexByHeader(r, drHeaders))
  .filter(report => _isoDateIST(report.date) === date);
```

BACKEND — E) _handleGetPlannedLessonForPeriod

Before (manual +5.5h math / split('T'))

After:

```
const selectedDateVal = plan.selectedDate || plan.date;
const planDate = _isoDateIST(selectedDateVal);
const queryDate = _normalizeQueryDate(date);
```

BACKEND — F) Timetable time combine helper (if start/end are time-only)

```
function _combineIST(dateYYYYMMDD, timeCell) {
  const dDate = _coerceToDate(dateYYYYMMDD) || new Date();
  const dTime = _coerceToDate(timeCell);
  if (!dTime) return null;

  const hh = Utilities.formatDate(dTime, TZ, 'HH');
  const mm = Utilities.formatDate(dTime, TZ, 'mm');
  const ss = Utilities.formatDate(dTime, TZ, 'ss');

  const base = new Date(Utilities.formatDate(dDate, TZ, 'yyyy-MM-dd') + 'T00:00:00');
  base.setHours(Number(hh), Number(mm), Number(ss), 0);
  return base; // IST datetime for that day+time
}
```

BACKEND — Project setting

Apps Script Editor → Project Settings → Time zone → set to **Asia/Kolkata**.
(We still keep TZ='Asia/Kolkata' in code to be explicit.)

FRONTEND — DailyReportTimetable.jsx (key normalization)

Normalize the key so period/class/subject match back-end exactly:

```
- function keyOf(r) {
-   return `${r.period}|${r.class}|${r.subject}`;
- }
+ function keyOf(r) {
+   const p   = String(Number(r.period));
+   const cls = String(r.class || '').trim();
+   const sub = String(r.subject || '').trim();
+   return `${p}|${cls}|${sub}`;
+ }
```

Also when mapping reports:

```
- const k = `${r.period}|${r.class}|${r.subject}`;
+ const k =
`${String(Number(r.period))}|${String(r.class||'').trim()}|${String(r.subject||'').trim()}`;
```

FRONTEND — Date discipline (IST YYYY-MM-DD)

Ensure your utils lock to IST:

```
// utils/dateUtils.js
export function todayIST() { /* returns YYYY-MM-DD */ }
export function formatLocalDate(yyyyMmDd) { /* pretty string */ }
export function periodToTimeString(period) { /* label map */ }
```

In component you already do:

```
const [date, setDate] = useState(todayIST());
// pass `date` unchanged to APIs; backend expects YYYY-MM-DD (IST).
```

FRONTEND — Unwrap API everywhere (timetable & reports)

Timetable unwrap:

```
- let ttList = [];
- if (Array.isArray(tt)) { ttList = tt; }
- else if (tt && Array.isArray(tt.periods)) { ttList = tt.periods; }
- else if (tt && tt.data && Array.isArray(tt.data)) { ttList = tt.data; }
+ let ttList = [];
+ const ttData = (tt && Array.isArray(tt.data)) ? tt.data : tt;
+ if (Array.isArray(ttData)) ttList = ttData;
+ else if (ttData && Array.isArray(ttData.periods)) ttList = ttData.periods;
```

Reports unwrap (you already have):

```
const reportsArray = Array.isArray(rep?.data) ? rep.data : (Array.isArray(rep) ? rep : []);
```

FRONTEND — Planned lesson field normalization

Server may send session vs sessionNo, learningObjectives vs objectives, etc.

```
// when auto-filling from planned lesson
- sessionNo: Number(lp.sessionNo || lp.session || 0),
- objectives: lp.learningObjectives || "",
- activities: lp.teachingMethods || "",
+ sessionNo: Number(lp.sessionNo ?? lp.session ?? 0),
+ objectives: lp.learningObjectives ?? lp.objectives ?? "",
+ activities: lp.teachingMethods ?? lp.activities ?? "",
  _session: lp.session ?? lp.sessionNo ?? ''

// when populating from submitted report
- sessionNo: Number(report.sessionNo || 0),
- objectives: report.objectives || "",
- activities: report.activities || "",
+ sessionNo: Number(report.sessionNo ?? report.session ?? 0),
+ objectives: report.objectives ?? "",
+ activities: report.activities ?? "",
  _session: report.session ?? report.sessionNo ?? ''
```

FRONTEND — Status defaults & Submit-All resilience

Mark non-mapped periods as not submitted (optional):

```
ttList.forEach(r => {
  const k = keyOf(r);
  if (!sm[k]) sm[k] = "Not Submitted";
});
setStatusMap({ ...sm });
```

Keep Submit-All moving even if one row fails:

```
for (const r of notSubmittedRows) {
  try {
    await handleSubmitOne(r);
    await new Promise(res => setTimeout(res, 150));
  } catch (e) {
    console.warn('Submit-one failed, continuing:', e);
  }
}
```

FRONTEND — Optional: switch between plans when multiple exist

```
{isPlanned && !submitted && (
  <div className="mt-2">
    <label className="text-xs text-gray-600">Choose lesson plan:</label>
    <select
      className="w-full text-xs border border-gray-300 rounded px-2 py-1.5 mt-1"
      value={d.lessonPlanId || ""}
      onChange={e => handleLessonPlanChange(k, e.target.value)}
    >
      <option value="">— Select —</option>
      {(lessonPlansMap[`r.class|{r.subject}`] || []).map(lp => (
        <option key={lp.lpId} value={lp.lpId}>
          {lp.chapter || 'Untitled'} {lp.session ? `(Session ${lp.session})` : ''}
        </option>
      ))}
    </select>
  </div>
)}
```

Checklist — What you must change (short)

1) Paste BACKEND helpers (TZ='Asia/Kolkata').
2) Apply A–F surgical edits (replace split('T') and manual math with helpers).
3) (Optional) Use _combineIST for time-only start/end.
4) Set Apps Script project timezone to Asia/Kolkata.
5) FRONTEND: normalize keyOf, unwrap API responses, normalize lesson fields, optional status default & submit-all backoff.