1. ⚙️ Setup & Installation

Backend

```
# clone repo
git clone git@github.com:charanrocky/Assessmentmanagementsystem.git
cd Assessmentmanagementsystem/backend

# install dependencies
npm install

# run in dev mode
npm run dev
```

Frontend

```
cd ../frontend

# install dependencies
npm install

# run in dev mode
npm run dev
```

2. 🏗️ Configuration System Design

The backend uses a **configuration-driven design**:

- `assessmentConfig.ts` → defines which *sections* exist for each assessment type.
- `fieldMappings.ts` → defines how report fields are pulled from raw `data.ts` (JSON path style).
- `classification.ts` → defines value ranges → categories (e.g., BMI = Normal/Overweight).
- `pdfService.ts` uses these configs dynamically → so **no code modification** is needed when new assessment types are added.

3. ➕ Adding a New Assessment Type

1. Open `src/config/assessmentConfig.ts`
2. Add a new entry:

```
export const assessmentConfig = {
  as_hr_02: { sections: ["Key Body Vitals", "Body Composition"] },
  as_card_01: { sections: ["Vitals", "Endurance"] },
  as_lung_01: { sections: ["Key Body Vitals", "Lung Capacity"] } //
👈 new type
};
```

3. Add data for `as_lung_01` inside `src/data/data.ts`.

✅ Done! Now the backend recognizes the new type.

4. 🔄 Modifying Field Mappings

Open **`src/config/fieldMappings.ts`**.

Example:

```
export const fieldMappings = {
  overallHealthScore: "accuracy",
  heartRate: "vitalsMap.vitals.heart_rate",
  bmi: "bodyCompositionData.BMI",
  endurance: "exercises[?(@.id==235)].setList[0].time"
};
```

Change Example

If a new dataset stores `heartRate` under `metadata.heart_scores.hr`, just update:

```
heartRate: "vitalsMap.metadata.heart_scores.hr"
```

✅ No code changes in controllers or services needed.

---

5. 📊 Updating Classification Ranges

Open **`src/config/classification.ts`**.

Example (BMI):

```
export const classification = {
```

```
  bmi: [
    { range: [0, 18.5], label: "Underweight" },
    { range: [18.5, 24.9], label: "Normal" },
    { range: [25, 29.9], label: "Overweight" },
    { range: [30, 100], label: "Obese" }
  ]
};
```

Change Example

If your medical standards define “Obese” as BMI > 32, just update:

```
{ range: [32, 100], label: "Obese" }
```

✅ All new reports will use updated classification automatically.

---

6. 📂 Example Config Files

assessmentConfig.ts

```
export const assessmentConfig = {
  as_hr_02: {
    sections: ["Key Body Vitals", "Heart Health", "Body
Composition"]
  },
  as_card_01: {
    sections: ["Vitals", "Cardiovascular Endurance", "Body
Composition"]
  }
};
```

---

fieldMappings.ts

```
export const fieldMappings = {
  overallHealthScore: "accuracy",
  heartRate: "vitalsMap.vitals.heart_rate",
  systolic: "vitalsMap.vitals.bp_sys",
  diastolic: "vitalsMap.vitals.bp_dia",
  bmi: "bodyCompositionData.BMI",
```

```
  endurance: "exercises[?(@.id==235)].setList[0].time"
};
```

---

classification.ts

```
export const classification = {
  bmi: [
    { range: [0, 18.5], label: "Underweight" },
    { range: [18.5, 24.9], label: "Normal" },
    { range: [25, 29.9], label: "Overweight" },
    { range: [30, 100], label: "Obese" }
  ],
  heartRate: [
    { range: [0, 59], label: "Low" },
    { range: [60, 100], label: "Normal" },
    { range: [101, 200], label: "High" }
  ]
};
```

✅ Summary

- **Setup**: Install → run backend + frontend.
- **Config system**: Uses `assessmentConfig`, `fieldMappings`, `classification`.
- **Add new assessment**: Update `assessmentConfig.ts` + add dataset.
- **Modify mappings**: Update JSON path in `fieldMappings.ts`.
- **Change classifications**: Update ranges in `classification.ts`.

This makes the system **100% extensible without code changes.**