🎯 TEAM RESPONSIBILITIES & DAILY TASKS

Gen Counselling AI for Good - Hackathon Production Plan

Timeline: January 26-31, 2026
Team Size: 10 members
Goal: Working prototype + demo by January 31st

📋 TABLE OF CONTENTS

🎨 FRONTEND TEAM (3 people)

FE-1: Forms & Validation Lead

Responsibilities:

Build all user input forms with client-side validation and data flow to backend.

Deliverables:

1. Landing Page

- Project introduction
- "Get Started" button → Registration

2. Registration/Basic Info Form

- Fields:
    - Age (number, 15-100)

  - Gender (dropdown: male/female/other)
  - Height (cm)
  - Weight (kg)
  - BMI (auto-calculated from height/weight)
- Validation:
  - All fields required
  - Reasonable ranges
- "Next" button → Lifestyle Form

3. Lifestyle Form

- Fields:
  - Smoking: Yes/No radio buttons
  - Alcohol: Dropdown (none/occasional/moderate/heavy)
  - Diet: Dropdown (balanced/high_sugar/high_fat_diet/high_salt)
  - Exercise: Dropdown (sedentary/occasional/regular)
  - Stress Level: Dropdown (low/moderate/high)
  - Sleep Hours: Number input (0-12)
- "Next" button → Family History

4. Family History Form (3 Generations - Array Format)

Generation 1 (Parents):

- Mother, Father
- Each has checkboxes for all 10 diseases

Generation 0 (Siblings - Dynamic):

- [+ Add Brother/Sister] button
- Can add multiple siblings
- Each has disease checkboxes

Generation -1 (Children - Optional):

- [+ Add Son/Daughter] button
- Only if user has children with diagnosed conditions
- Rarely needed, but important for hereditary diseases

Generation 2 (Extended Family):

- Fixed: 4 Grandparents (maternal/paternal grandmother/grandfather)
- Dynamic: [+ Add Aunt/Uncle] buttons
- Can add multiple aunts/uncles per side

Data Structure:

javascript

```javascript
const familyHistory = [
  {
   role: "mother",
   generation: 1,
   type2_diabetes: true,
   hypertension: true
  },
  {
   role: "sister",
   generation: 0,
   pcos: true
  },
  {
   role: "maternal_aunt",
   generation: 2,
   breast_ovarian_cancer: true
  },
  {
   role: "maternal_aunt",  // Multiple aunts allowed!
   generation: 2,
   breast_ovarian_cancer: true
  }
];
```

- "Next" button → Upload Report (optional) or "Skip to Results"

5. Data Conversion to JSON (Array Format)

javascript

```
const formData = {
  basic_info: {
    age: 32,
    gender: "female",
    height: 162,
    weight: 78,
    bmi: 29.7
  },
  lifestyle: {
    smoking: false,
    alcohol: "occasional",
    exercise: "sedentary",
    diet: "high_sugar",
    sleep_hours: 6,
    stress_level: "high"
  },
  family_history: [
    {
      role: "mother",
      generation: 1,
      type2_diabetes: true,
      hypertension: true
    },
    {
      role: "father",
      generation: 1,
      cad: true
    },
    {
      role: "sister",
      generation: 0,
      type2_diabetes: true
    },
    {
      role: "maternal_grandmother",
      generation: 2,
      type2_diabetes: true
    },
    {
      role: "maternal_aunt",
      generation: 2,
      breast_ovarian_cancer: true
    },
    {
      role: "maternal_aunt",
      generation: 2,
      breast_ovarian_cancer: true
    }
  ],
  lab_values: {}
};
```

6. Send to Backend

```javascript
const response = await fetch('http://localhost:8000/predict-risk', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(formData)
});
```

Tech Stack:

- React
- React Hook Form (for validation)
- Tailwind CSS

Timeline:

- **Day 1 (26th):** Wireframe forms, setup components
- **Day 2 (27th):** ✅ All forms working, validation complete, data flows to backend
- **Day 3+:** Polish UI, add transitions

FE-2: Results Dashboard Lead

Responsibilities:

Display ranked disease predictions in a clean, understandable dashboard.

Deliverables:

1. Results Dashboard Page

Layout:

```
┌─────────────────────────────────────────┐
│ Your Health Risk Assessment Results     │
│                                         │
│  ┌───────────────────────────────────┐  │
│  │ Risk Class Legend                 │  │
│  │ 🟢 I: Low  🟡 II: Moderate        │  │
│  │ 🟠 III: High  🔴 IV: Very High    │  │
│  └───────────────────────────────────┘  │
│                                         │
│  ┌───────────────────────────────────┐  │
│  │ Disease Name      │ Risk │ Class  │  │
│  ├───────────────────────────────────┤  │
│  │ Type-2 Diabetes   │ 68%  │ 🟠 III │  │
│  │ Hypertension      │ 52%  │ 🟡 II  │  │
│  │ CAD               │ 45%  │ 🟡 II  │  │
│  │ ...               │ ...  │ ...    │  │
│  └───────────────────────────────────┘  │
└─────────────────────────────────────────┘
```

Components to Build:

a) Risk Class Legend

- 4 colored boxes showing I-IV
- Label: Low, Moderate, High, Very High
- Color codes:
  - Class I: 🟩 #22c55e (Green)
  - Class II: 🟨 #eab308 (Yellow)
  - Class III: 🟧 #f97316 (Orange)
  - Class IV: 🟥 #ef4444 (Red)

b) Results Table

- Columns:
  - Disease Name (clickable → detail page)
  - Probability (percentage)
  - Risk Class (badge with color)
  - Urgency (badge: None/Routine/Soon/Urgent)
- Sorted by probability (highest first)
- Click row → navigate to disease detail page

c) Optional: Chart Visualization

- Bar chart of top 5 risks using Recharts
- Color bars by risk class

d) Loading State

- Spinner while waiting for API response
- "Analyzing your health data..."

e) Error Handling

- If API fails: "Unable to load results. Please try again."
- Retry button

2. Data Processing

javascript

```javascript
const [results, setResults] = useState([]);

useEffect(() => {
  // After form submission
  fetch('/predict-risk', { method: 'POST', body: formData })
    .then(res => res.json())
    .then(data => {
      setResults(data.results); // Array of disease objects
    })
    .catch(err => {
      // Show error state
    });
}, []);
```

Tech Stack:

- React
- Recharts (for charts)
- Tailwind CSS
- React Router (for navigation)

Timeline:

- **Day 2 (27th):** Build dashboard skeleton with dummy data
- **Day 3 (28th):** ✅ Connect to real API, display actual results
- **Day 4+:** Add chart, polish UI

FE-3: Disease Detail & Coach UI Lead

Responsibilities:

Build disease detail pages and OCR upload component.

Deliverables:

1. Disease Detail Page

Layout:

```
┌─────────────────────────────────────────────┐
│ ← Back to Dashboard                          │
│                                             │
│ Type-2 Diabetes                             │
│ 🟠 Risk Class III - High Risk               │
│                                             │
│ ──────────────────────────────────────────── │
│                                             │
│ 📋 What This Means                          │
│ A chronic condition where the body...       │
│                                             │
│ ⚠️ Why You're At Risk                       │
│ • Mother has this condition                 │
│ • HbA1c elevated at 6.2%...                 │
│ • High sugar intake...                      │
│ • Sedentary lifestyle...                    │
│                                             │
│ 💪 Prevention Plan                          │
│ • Eliminate sugary drinks immediately       │
│ • Reduce sugar and refined carbs            │
│ • 30 minutes moderate activity daily        │
│ • Control portion sizes                     │
│                                             │
│ 🩺 Recommended Tests                        │
│ • HbA1c (Glycated Hemoglobin)               │
│ • Fasting Blood Glucose                     │
│ • Oral Glucose Tolerance Test               │
│                                             │
│ 🧑‍⚕️ Doctor Consultation                      │
│ Urgency: Soon (within 4-6 weeks)            │
│ Specialist: Endocrinologist                 │
│ What to discuss:                            │
│ • Blood sugar levels and HbA1c results      │
│ • Diet plan and carb management             │
│                                             │
└─────────────────────────────────────────────┘
```

Components:

a) Disease Header

- Disease name
- Risk class badge with color
- Description

b) Reasons Section

- Icon: ⚠️
- Title: "Why You're At Risk"
- Bullet list from `reasons` array

c) Prevention Section

- Icon: 💪
- Title: "Your Prevention Plan"
- Bullet list from `prevention` array

d) Tests Section

- Icon: 🩺
- Title: "Recommended Screening Tests"
- List from `recommended_tests` array

e) Consultation Section

- Icon: 🧑‍⚕️
- Title: "Doctor Consultation"
- Display `consult_detail`:
  - Urgency level with color
  - Timeframe
  - Recommended specialist
  - What to discuss (bullet list)

2. Upload Report Component (OCR)

Layout:

```
┌─────────────────────────────────────┐
│ Upload Lab Report (Optional)        │
│                                     │
│ ┌─────────────────────────────────┐ │
│ │                                 │ │
│ │   📄 Drag & drop report here    │ │
│ │   or click to browse            │ │
│ │                                 │ │
│ │   Supports: PDF, PNG, JPG       │ │
│ └─────────────────────────────────┘ │
│                                     │
│ [Extract Lab Values] button         │
│                                     │
│ Extracted Values:                   │
│ • HbA1c: 6.2%                       │
│ • Fasting Glucose: 115 mg/dL        │
│ • LDL: 145 mg/dL                    │
│                                     │
│ [Continue to Results] button        │
└─────────────────────────────────────┘
```

Functionality:

```javascript
const handleUpload = async (file) => {
  const formData = new FormData();
  formData.append('file', file);

  const response = await fetch('/ocr', {
    method: 'POST',
    body: formData
  });

  const { lab_values } = await response.json();
  // Add to user_data.lab_values
  // Show extracted values to user
};
```

3. UI Polish

- Navigation breadcrumbs
- Smooth transitions between pages
- Mobile responsive design
- Loading states for all async operations

- Success/error toast messages

Tech Stack:

- React
- Lucide React (icons)
- React Dropzone (file upload)
- Tailwind CSS

Timeline:

- **Day 3 (28th):** Build upload component, basic detail page
- **Day 4 (29th):** ✅ Complete all detail sections, polish UI
- **Day 5 (30th):** Final responsive polish

⚙️ BACKEND TEAM (3 people)

BE-1: Backend Lead + Database

Responsibilities:

Set up FastAPI application, database (if needed), and core infrastructure.

Deliverables:

1. FastAPI Application Setup

Project Structure:

```
backend/
├── main.py            # FastAPI app
├── models.py          # Pydantic models
├── database.py        # DB connection (optional)
├── config.py          # Environment config
└── requirements.txt   # Dependencies
```

main.py:

```python
```

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI(title="Gen Counselling API")

# CORS configuration
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Frontend URL
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/health")
async def health_check():
    return {"status": "healthy"}

@app.get("/")
async def root():
    return {"message": "Gen Counselling API v1.0"}
```

2. Database Schema (Optional for MVP)

If storing user data:

python

```
# models.py (SQLAlchemy)
from sqlalchemy import Column, Integer, String, JSON, DateTime
from database import Base

class UserProfile(Base):
    __tablename__ = "user_profiles"

    id = Column(Integer, primary_key=True)
    age = Column(Integer)
    gender = Column(String)
    basic_info = Column(JSON)
    lifestyle = Column(JSON)
    family_history = Column(JSON)
    created_at = Column(DateTime)

class PredictionHistory(Base):
    __tablename__ = "predictions"

    id = Column(Integer, primary_key=True)
    user_id = Column(Integer)
    results = Column(JSON)
    created_at = Column(DateTime)
```

3. Pydantic Models for Validation

python

```
from pydantic import BaseModel
from typing import Dict, Optional

class BasicInfo(BaseModel):
    age: int
    gender: str
    height: float
    weight: float
    bmi: float

class Lifestyle(BaseModel):
    smoking: bool
    alcohol: str
    exercise: str
    diet: str
    sleep_hours: float
    stress_level: str

class UserData(BaseModel):
    basic_info: BasicInfo
    lifestyle: Lifestyle
    family_history: Dict
    lab_values: Optional[Dict] = {}
```

4. Environment Configuration

python

```
# config.py
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    DATABASE_URL: str = "sqlite:///./app.db"
    CORS_ORIGINS: list = ["http://localhost:3000"]

    class Config:
        env_file = ".env"

settings = Settings()
```

5. CORS Setup & Testing

- Test from frontend that requests work
- Handle preflight OPTIONS requests

Tech Stack:

- FastAPI
- SQLAlchemy (if using DB)
- Pydantic
- Uvicorn (ASGI server)

Timeline:

- **Day 1 (26th):** FastAPI skeleton, health endpoint
- **Day 2 (27th):** ✅ CORS working, can receive requests from frontend
- **Day 3+:** Add database if needed, deploy backend

BE-2: OCR Endpoint + Upload Handling

Responsibilities:

Handle file uploads and integrate with AI-1's OCR module.

Deliverables:

1. File Upload Endpoint

python

```
from fastapi import UploadFile, File, HTTPException
import shutil
from pathlib import Path

@app.post("/ocr")
async def extract_labs_from_report(file: UploadFile = File(...)):
    """
    Extract lab values from uploaded medical report

    Accepts: PNG, JPG, PDF
    Returns: Dict of lab values
    """

    # 1. Validate file type
    allowed_types = ["image/png", "image/jpeg", "application/pdf"]
    if file.content_type not in allowed_types:
        raise HTTPException(400, "Invalid file type")

    # 2. Validate file size (max 10MB)
    file_size = 0
    chunk_size = 1024 * 1024  # 1MB
    temp_file = Path(f"temp/{file.filename}")

    with temp_file.open("wb") as buffer:
        while chunk := await file.read(chunk_size):
            file_size += len(chunk)
            if file_size > 10 * 1024 * 1024:  # 10MB
                raise HTTPException(400, "File too large")
            buffer.write(chunk)

    # 3. Call AI-1's OCR module
    try:
        from ai.ocr.ocr_pipeline import extract_labs_from_report

        lab_values = extract_labs_from_report(str(temp_file))

        # Clean up temp file
        temp_file.unlink()

        return {
            "success": True,
            "lab_values": lab_values
        }

    except Exception as e:
        # Clean up on error
        if temp_file.exists():
            temp_file.unlink()

        raise HTTPException(500, f"OCR failed: {str(e)}")
```

2. Error Handling

- File type validation
- File size limits
- OCR failure fallback
- Temp file cleanup

3. Response Format

```json
{
  "success": true,
  "lab_values": {
    "hba1c": 6.2,
    "fasting_glucose": 115,
    "ldl": 145,
    "hdl": 38,
    "triglycerides": 180,
    "systolic_bp": 135,
    "diastolic_bp": 88
  }
}
```

4. Coordinate with AI-1

- Ensure AI-1's function returns dict with normalized keys
- Handle cases where OCR can't extract certain values
- Test with multiple report formats

Tech Stack:

- FastAPI
- Python-Multipart (file handling)
- Pillow (image processing)

Timeline:

- **Day 2 (27th):** File upload endpoint skeleton

- **Day 3 (28th):** ✅ OCR integration working, returns structured labs
- **Day 4+:** Error handling, multiple file format support

BE-3: Prediction + Disease Info Endpoints

Responsibilities:

Main prediction endpoint that calls AI-2's risk model.

Deliverables:

1. Main Prediction Endpoint

```python
from fastapi import HTTPException
from ai.risk.risk_model import predict_risks

@app.post("/predict-risk")
async def predict_disease_risk(user_data: dict):
    """
    Predict disease risks based on user data

    Input: User profile (basic_info, lifestyle, family_history, lab_values)
    Output: Ranked list of disease predictions
    """


    try:
        # Validate input
        if not user_data.get("basic_info"):
            raise HTTPException(400, "basic_info required")

        if not user_data.get("lifestyle"):
            raise HTTPException(400, "lifestyle required")

        # Call AI-2 risk model
        results = predict_risks(user_data)

        return {
            "success": True,
            "results": results
        }

    except Exception as e:
        raise HTTPException(500, f"Prediction failed: {str(e)}")
```

2. Request Validation with Pydantic

python

```python
from pydantic import BaseModel, validator
from typing import Dict, Optional

class UserDataRequest(BaseModel):
    basic_info: Dict
    lifestyle: Dict
    family_history: Dict
    lab_values: Optional[Dict] = {}

    @validator('basic_info')
    def validate_basic_info(cls, v):
        required = ['age', 'gender', 'bmi']
        if not all(k in v for k in required):
            raise ValueError(f"basic_info must contain: {required}")
        return v

@app.post("/predict-risk")
async def predict(user_data: UserDataRequest):
    results = predict_risks(user_data.dict())
    return {"results": results}
```

3. Optional: Disease Info Endpoint

python

```
import json
from pathlib import Path

@app.get("/disease-info/{disease_id}")
async def get_disease_info(disease_id: str):
    """Get detailed information about a specific disease"""

    # Load from config
    config_path = Path("ai/data/diseases_config.json")
    with open(config_path) as f:
        config = json.load(f)

    disease = next(
        (d for d in config["diseases"] if d["id"] == disease_id),
        None
    )

    if not disease:
        raise HTTPException(404, "Disease not found")

    return disease
```

4. Error Handling

python

```
from fastapi import Request
from fastapi.responses import JSONResponse

@app.exception_handler(Exception)
async def global_exception_handler(request: Request, exc: Exception):
    return JSONResponse(
        status_code=500,
        content={
            "success": False,
            "error": str(exc),
            "detail": "Internal server error"
        }
    )
```

5. Response Format

json

```
{
  "success": true,
  "results": [
    {
      "disease_name": "Type-2 Diabetes",
      "disease_id": "type2_diabetes",
      "probability": 0.68,
      "risk_class": "III",
      "reasons": ["...", "...", "..."],
      "prevention": ["...", "...", "..."],
      "recommended_tests": ["...", "..."],
      "consult": "soon",
      "consult_detail": { /* full object */ }
    }
    // ... 9 more diseases
  ]
}
```

6. Coordinate with AI-2

- Test import: `from ai.risk.risk_model import predict_risks`
- Handle edge cases (missing data)
- Test with demo cases

Tech Stack:

- FastAPI
- Pydantic

Timeline:

- **Day 2 (27th):** Endpoint skeleton, test with dummy data
- **Day 3 (28th):** ✅ AI-2 integration working, returns real predictions
- **Day 4+:** Add caching, optimize performance

🤖 AI TEAM (2 people)

AI-1: OCR & Lab Extraction Lead

Responsibilities:

Extract lab values from uploaded medical reports using OCR.

Deliverables:

1. OCR Pipeline (`ai/ocr/ocr_pipeline.py`)

```python
import easyocr
import re
from pathlib import Path

def extract_labs_from_report(image_path: str) -> dict:
    """
    Main OCR function called by backend

    Args:
        image_path: Path to uploaded report image/PDF

    Returns:
        Dict of normalized lab values
    """

    # Initialize OCR reader
    reader = easyocr.Reader(['en'])

    # Extract text
    result = reader.readtext(image_path)
    full_text = " ".join([item[1] for item in result])

    # Parse lab values
    lab_values = parse_lab_values(full_text)

    # Normalize units
    lab_values = normalize_units(lab_values)

    return lab_values
```

2. Lab Parser (`ai/ocr/report_parser.py`)

```python
```

```
import re

def parse_lab_values(text: str) -> dict:
    """Extract lab values using regex patterns"""

    lab_values = {}

    # HbA1c pattern: "HbA1c: 6.2" or "HbA1c 6.2%"
    hba1c_match = re.search(
        r'HbA1c[:\s]+(\d+\.?\d*)',
        text,
        re.IGNORECASE
    )
    if hba1c_match:
        lab_values['hba1c'] = float(hba1c_match.group(1))

    # Fasting Glucose
    glucose_match = re.search(
        r'(?:Fasting\s+)?Glucose[:\s]+(\d+)',
        text,
        re.IGNORECASE
    )
    if glucose_match:
        lab_values['fasting_glucose'] = float(glucose_match.group(1))

    # LDL Cholesterol
    ldl_match = re.search(
        r'LDL[:\s]+(\d+)',
        text,
        re.IGNORECASE
    )
    if ldl_match:
        lab_values['ldl'] = float(ldl_match.group(1))

    # HDL Cholesterol
    hdl_match = re.search(
        r'HDL[:\s]+(\d+)',
        text,
        re.IGNORECASE
    )
    if hdl_match:
        lab_values['hdl'] = float(hdl_match.group(1))

    # Triglycerides
    tg_match = re.search(
        r'Triglycerides[:\s]+(\d+)',
        text,
        re.IGNORECASE
    )
    if tg_match:
        lab_values['triglycerides'] = float(tg_match.group(1))

    # Blood Pressure (if present)
    bp_match = re.search(
        r'(\d{2,3})/(\d{2,3})',
        text
    )
    if bp_match:
        lab_values['systolic_bp'] = float(bp_match.group(1))
        lab_values['diastolic_bp'] = float(bp_match.group(2))

    return lab_values
```

3. Unit Normalizer (`ai/ocr/normalize_units.py`)

```python
def normalize_units(lab_values: dict) -> dict:
    """
    Normalize units to standard format

    - HbA1c: % (already correct)
    - Glucose: mg/dL (convert from mmol/L if needed)
    - Cholesterol: mg/dL (convert from mmol/L if needed)
    """

    normalized = {}

    for key, value in lab_values.items():
        if key == 'fasting_glucose':
            # If value < 20, likely mmol/L, convert to mg/dL
            if value < 20:
                value = value * 18.0  # mmol/L to mg/dL

        elif key in ['ldl', 'hdl', 'triglycerides', 'total_cholesterol']:
            # If value < 10, likely mmol/L
            if value < 10:
                value = value * 38.67  # mmol/L to mg/dL (approx)

        normalized[key] = value

    return normalized
```

4. Sample Report Testing Create 2-3 sample medical reports with varying formats:

- Format 1: Standard lab format
- Format 2: Handwritten-style
- Format 3: Different layout

Test OCR accuracy on each.

5. Error Handling

```python
def extract_labs_from_report(image_path: str) -> dict:
    try:
        # OCR logic
        pass
    except Exception as e:
        # Log error
        print(f"OCR failed: {e}")
        # Return empty dict or partial results
        return {}
```

Tech Stack:

- EasyOCR (preferred) or Tesseract
- OpenCV (image preprocessing)
- Pillow (image handling)
- Regex (pattern matching)

Timeline:

- **Day 2 (27th):** OCR extracts text from image
- **Day 3 (28th):** ✅ Parser extracts lab values, normalizer works
- **Day 4+:** Test multiple formats, improve accuracy

AI-2: Risk Prediction & Coaching Lead ✅

Status: COMPLETE

All modules built and documented:

- ✅ `risk_model.py` - Main prediction function
- ✅ `scoring_rules.py` - Family/lifestyle/lab scoring

- ✅ `risk_classes.py` - Risk classification
- ✅ `explainability.py` - Reason generation
- ✅ `prevention_engine.py` - Prevention recommendations
- ✅ `test_recommender.py` - Test suggestions
- ✅ `consult_logic.py` - Consultation urgency
- ✅ All JSON configs

Current Tasks:

Day 2 (27th) - TODAY:

1. **Test Module**

```bash
cd ai/
python3 -c "
from risk.risk_model import predict_risks
import json

# Load demo case
with open('data/sample_inputs.json') as f:
    demos = json.load(f)['demo_cases']

# Test prediction
result = predict_risks(demos['case_b_high_diabetes_risk'])

# Print top result
print(json.dumps(result[0], indent=2))
"
```

2. **Create `__init__.py` files:**

```python
# ai/__init__.py
from .risk.risk_model import predict_risks

# ai/risk/__init__.py
from .risk_model import predict_risks

# ai/coaching/__init__.py
(can be empty)
```

3. **Coordinate with BE-3:**
    - Ensure they can import `predict_risks()`
    - Test with demo case together
    - Fix any import errors
4. **Coordinate with AI-1:**
    - Confirm lab key format from OCR
    - Test that OCR output works with your scoring

Day 3 (28th):

- Test integration: OCR labs → predict_risks()
- Fix any edge cases
- Optimize if needed

Day 4+ (29-30th):

- Add more demo cases if needed
- Fine-tune prevention recommendations
- Help with integration debugging

No Additional Code Needed

Your module is production-ready! Focus on testing and integration.

🔗 API LEAD (1 person)

Responsibilities:

Ensure all teams communicate through standardized API contracts. Act as integration glue.

Deliverables:

1. API Contract Document

Create shared doc (Google Doc / Notion) with:

Endpoint 1: POST /predict-risk (NEW HYBRID FORMAT)

```
URL: POST http://localhost:8000/predict-risk
Content-Type: application/json

Request Body:
{
  "patient": {
    "age": 32,
    "gender": "F",
    "weight": 78.5,
    "height": 162,
    "race": "asian",
    "known_issues": []
  },
  "lifestyle": {
    "smoking": false,
    "alcohol": "occasional",
    "exercise": "sedentary",
    "diet": "high_sugar",
    "sleep_hours": 6.5,
    "stress_level": "high"
  },
  "family": [
    {
      "role": "mother",
      "generation": 1,
      "age": 58,
      "gender": "F",
      "weight": 65.0,
      "height": 160,
      "race": "asian",
      "known_issues": ["type2_diabetes", "hypertension"]
    },
    {
      "role": "sister",
      "generation": 0,
      "age": 28,
      "gender": "F",
      "known_issues": ["type2_diabetes"]
    },
    {
      "role": "maternal_grandmother",
      "generation": 2,
      "age": 82,
      "gender": "F",
      "known_issues": ["type2_diabetes"]
    }
  ],
  "lab_values": {
    "hba1c": 6.2,
    "fasting_glucose": 115,
    "ldl": 145,
    "hdl": 38
  }
}

Response (200 OK):
{
  "success": true,
  "results": [
    {
      "disease_name": "Type-2 Diabetes",
      "disease_id": "type2_diabetes",
      "probability": 0.82,
      "risk_class": "IV",
      "reasons": [
        "Mother has this condition",
        "Sister has this condition",
        "Maternal Grandmother has this condition",
        "HbA1c elevated at 6.2%, suggesting higher diabetes risk",
        "High sugar intake and poor dietary habits"
      ],
      "prevention": ["Eliminate sugary drinks", "..."],
      "recommended_tests": ["HbA1c", "..."],
      "consult": "urgent"
    }
  ]
}
```

Endpoint 2: POST /ocr

```
URL: POST http://localhost:8000/ocr
Content-Type: multipart/form-data

Request: FormData with file upload
{
  file: <binary data>
}

Response (200 OK):
{
  "success": true,
  "lab_values": {
    "hba1c": 6.2,
    "fasting_glucose": 115,
    "ldl": 145,
    "hdl": 38
  }
}
```

Endpoint 3: GET /health

```
URL: GET http://localhost:8000/health

Response (200 OK):
{
  "status": "healthy"
}
```

2. Postman Collection

Create collection with:

- All 3 endpoints
- Sample requests with valid data
- Environment variables (API_URL = http://localhost:8000)
- Share JSON export with team

3. Daily Integration Testing Checklist

Day 2:

□ Backend `/health` endpoint responds
□ Frontend can call backend (CORS working)
□ `/predict-risk` accepts JSON (even with dummy response)

Day 3:

□ `/ocr` accepts file upload
□ `/predict-risk` returns real predictions from AI-2
□ Frontend displays results from backend
□ Full flow: Form → Backend → AI → Frontend

Day 4:

□ Upload → OCR → Extract labs → Predict → Display
□ Error handling works (missing data, API failures)
□ All edge cases handled

Day 5:

□ Performance testing (response times)
□ Cross-browser testing
□ Mobile responsive testing

4. Handle Integration Issues

Common issues to fix:

- **CORS errors:** Ensure backend allows frontend origin
- **JSON format mismatches:** Frontend sends wrong structure
- **Missing fields:** Validation errors
- **File upload issues:** Wrong content-type or file format
- **API timeouts:** Long OCR processing

5. Maintain Bug List

Track in shared doc:

Priority	Issue	Owner	Status
Critical	CORS blocking requests	BE-1	Fixed
High	OCR timeout on large files	BE-2	In Progress
Medium	UI not showing all diseases	FE-2	Open

Tech Stack:

- Postman (API testing)
- Browser DevTools (debugging)

- Shared documentation (Google Docs)

Timeline:

- **Day 1 (26th):** ✅ API contract written, shared with all teams
- **Day 2 (27th):** Postman collection created, test basic endpoints
- **Day 3-5 (28-30th):** Daily integration testing, bug fixes
- **Day 6 (31st):** Final integration verification

📊 PRESENTATION LEAD (1 person)

Responsibilities:

Create all presentation materials: PPT, demo video, reflection video, and demo script.

Deliverables:

1. PowerPoint Presentation (15-20 slides)

Slide Breakdown:

Slide 1: Title

- Project name: "Gen Counselling AI for Good"
- Tagline: "Preventive Health Through AI-Powered Risk Awareness"
- Team name
- Hackathon name

Slides 2-3: Problem Statement

- Statistics on late diagnosis of hereditary diseases
- Impact: healthcare costs, preventable deaths
- Visual: Chart showing diagnosis delays
- Real story: Peer with undiagnosed diabetes despite family history

Slide 4: Target Users

- Demographics: 15-55 years, urban/semi-urban
- Use cases:
  - Student with family history of diabetes
  - Working professional avoiding expensive genetic tests
  - Middle-aged person wanting preventive screening

Slides 5-6: Our Solution

- What we built: Web-based AI platform
- How it works: Input → AI Analysis → Risk Assessment → Guidance
- Flowchart showing user journey

Slide 7: AI Innovation - OCR

- Upload medical reports
- AI extracts lab values automatically
- Makes health data accessible without typing

Slide 8: AI Innovation - Risk Prediction

- Analyzes family history (2 generations)
- Considers lifestyle factors
- Processes lab biomarkers
- Output: Ranked disease risks (I-IV)

Slide 9: AI Innovation - Health Coaching

- Personalized prevention plans
- Screening test recommendations
- Doctor consultation guidance
- Acts as continuous health coach

Slides 10-14: Prototype Walkthrough

- Slide 10: Landing page screenshot
- Slide 11: Family history form screenshot
- Slide 12: Upload report screenshot
- Slide 13: Results dashboard screenshot
- Slide 14: Disease detail page screenshot

Slides 15-16: Demo Cases

- **Case A (Low Risk):**
    - Profile: 24-year-old, no family history, healthy habits
    - Result: All diseases Class I-II
    - Action: Maintain healthy lifestyle

- **Case B (High Diabetes Risk):**
  - Profile: 32-year-old, mother diabetic, high sugar diet, HbA1c 6.2%
  - Result: Diabetes Class III (68% risk)
  - Reasons: Mother affected, elevated HbA1c, poor diet, sedentary
  - Action: Consult endocrinologist soon, lifestyle changes

Slide 17: Impact & SDG Alignment

- SDG 3: Good Health and Well-Being
- Early intervention → Reduced disease burden
- Accessible screening → Health equity
- Scalability: Can reach millions

Slide 18: Future Roadmap

- Integration with hospital systems
- Multi-language support
- ML model training on population data
- Mobile app version
- Telemedicine integration

Slide 19: Ethics & Privacy

- No data storage
- Non-diagnostic tool (recommends professional consultation)
- Transparent AI reasoning
- Responsible genetic information handling

Slide 20: Thank You + Q&A

- Team contact info
- GitHub link
- Demo link

Design Tips:

- Use consistent color scheme (match risk class colors)
- Minimal text per slide (max 3 bullet points)
- High-quality screenshots
- Icons from Lucide/Heroicons

- Professional fonts (Inter, Poppins)

---

2. Demo Video (2-3 minutes)

Script:

[0:00-0:20] Problem Introduction

- Video: Text overlay on health-related visuals
- Voiceover: "Every year, millions are diagnosed late with hereditary diseases. Diabetes, heart disease, cancer—these run in families, but most people don't know their risk until it's too late."

[0:20-0:40] Solution Overview

- Video: Show landing page
- Voiceover: "Introducing Gen Counselling AI—a preventive health platform that predicts your inherited disease risk using family history, lifestyle, and medical records."

[0:40-2:00] Prototype Walkthrough

- **[0:40-0:55]** Fill out form
  - Screen recording: Enter age, select diet, mark mother has diabetes
  - Voiceover: "Simply enter your basic info, lifestyle habits, and family medical history."
- **[0:55-1:10]** Upload report
  - Screen recording: Drag-drop lab report, click "Extract Labs"
  - Voiceover: "Upload your lab reports—our AI extracts the values automatically."
- **[1:10-1:30]** View results
  - Screen recording: Dashboard shows ranked diseases
  - Voiceover: "Within seconds, you get your personalized risk assessment. Diabetes is flagged as high risk—Class III."
- **[1:30-2:00]** Disease detail
  - Screen recording: Click diabetes, scroll through reasons, prevention, tests
  - Voiceover: "The system explains why you're at risk, gives personalized prevention tips, recommends screening tests, and tells you when to see a doctor."

[2:00-2:30] AI Capabilities

- Video: Split screen showing AI features
- Voiceover: "Powered by AI: OCR for medical records, risk prediction across 10 diseases, and intelligent health coaching tailored to you."

[2:30-3:00] Impact Statement

- Video: Happy people, health icons
- Voiceover: "Early awareness saves lives. Gen Counselling AI brings preventive healthcare to everyone—no expensive genetic tests needed. Together, we can shift from treatment to prevention."

Production:

- Use screen recording tool (OBS, Loom)
- Add background music (royalty-free from YouTube Audio Library)
- Use text overlays for key points
- Export: 1080p MP4

---

3. Reflection Video (2-3 minutes)

Content:

Team Introduction (0:00-0:30)

- Each member introduces themselves
- "Hi, I'm [Name], and I worked on [Frontend/Backend/AI]."

Inspiration (0:30-1:00)

- "We were inspired by a peer who was diagnosed with diabetes late despite having a strong family history. We realized how many people lack awareness of inherited risks."

Challenges & Solutions (1:00-1:45)

- **Challenge 1:** "Integrating OCR with diverse report formats was tricky. We solved it by creating regex patterns and normalizing units."
- **Challenge 2:** "Explaining AI predictions in simple language. We built an explainability engine that generates human-readable reasons."
- **Challenge 3:** "Coordinating 10 people across frontend, backend, and AI. Our API lead created a contract document that kept everyone aligned."

Key Learnings (1:45-2:15)

- "We learned the importance of early integration testing."
- "We discovered how powerful AI can be for preventive health, not just diagnosis."
- "Teamwork and clear communication were critical to building this in 6 days."

What Makes It Unique (2:15-2:45)

- "Unlike genetic testing services that cost thousands, our platform is accessible to anyone with a smartphone."
- "We combine AI-OCR, risk prediction, and health coaching in one seamless experience."
- "Our focus on prevention, not just diagnosis, sets us apart."

Closing (2:45-3:00)

- "We're proud of what we built, and we hope it can make a real impact on preventive healthcare globally."

Production:

- Record via Zoom or Google Meet
- Each team member speaks for 20-30 seconds
- Edit together with transitions
- Add team photo at end

---

4. Demo Script for Live Presentation

Setup:

- Have 2 browser tabs open:
  - Tab 1: Case A (Low Risk) - already filled
  - Tab 2: Case B (High Diabetes Risk) - ready to demo

Script (3 minutes):

[0:00-0:15] Introduction "Hi everyone, I'm [Name] and we built Gen Counselling AI—a platform that predicts your inherited disease risk and guides prevention."

[0:15-0:45] Problem "Many people unknowingly carry genetic predispositions. By the time symptoms appear, treatment options are limited. Our platform changes that with early awareness."

[0:45-1:30] Demo - Case B

- "Let me show you. This is Sarah, 32 years old. Her mother has diabetes."
- [Fill form quickly, highlighting key fields]
- "She has a high-sugar diet, sedentary lifestyle."
- [Upload report]
- "She uploads her recent lab report—AI extracts HbA1c of 6.2%."

- [Click "Predict"]
- "Within seconds..."

[1:30-2:15] Results

- [Dashboard appears]
- "Sarah's at 68% risk for Type-2 Diabetes—Class III, high risk."
- [Click diabetes]
- "The system explains why: her mother's history, elevated HbA1c, poor diet, lack of exercise."
- "It gives personalized prevention tips: eliminate sugary drinks, exercise 30 minutes daily."
- "Recommends tests: HbA1c, fasting glucose."
- "And tells her: consult an endocrinologist within 4-6 weeks."

[2:15-2:45] AI Highlight "This is powered by AI across three layers: OCR for medical records, risk prediction across 10 diseases, and intelligent health coaching. All without expensive genetic testing."

[2:45-3:00] Impact "Early awareness saves lives. We built this to bring preventive healthcare to everyone. Thank you!"

Backup Plan:

- If live demo fails: Show demo video instead
- Have screenshots ready as backup

Tech Stack:

- PowerPoint / Google Slides
- Screen recording: OBS Studio / Loom
- Video editing: DaVinci Resolve (free) / iMovie
- Audio: Audacity (free)

Timeline:

- **Day 1-2 (26-27th):** Slide outline, collect content
- **Day 3 (28th):** Draft slides, collect screenshots
- **Day 4 (29th):** ✅ 60% slides done, demo script written
- **Day 5 (30th):** ✅ 100% slides, demo video recorded
- **Day 6 (31st):** Final rehearsal, reflection video done

🧪 OPTIONAL: QA/TESTER

Responsibilities:

Test the entire system end-to-end and maintain bug list.

Deliverables:

1. Test Cases

Happy Path:

☐ Landing page loads
☐ Can fill all forms
☐ Can submit without OCR (manual entry only)
☐ Dashboard shows 10 diseases ranked
☐ Can click disease to see detail
☐ All sections display (reasons, prevention, tests, consult)

OCR Path:

☐ Can upload PNG file
☐ Can upload JPG file
☐ Can upload PDF file
☐ OCR extracts values correctly
☐ Extracted values displayed to user
☐ Prediction uses OCR data

Edge Cases:

☐ Submit form with no family history → Works
☐ Submit without lab values → Works
☐ Upload invalid file type → Shows error
☐ Upload huge file (>10MB) → Shows error
☐ Backend down → Shows friendly error
☐ OCR fails → Graceful fallback

Browser Testing:

☐ Chrome (desktop)
☐ Firefox (desktop)
☐ Safari (desktop)
☐ Chrome (mobile)
☐ Safari (iOS mobile)

2. Bug Report Template

ID	Priority	Issue	Steps to Reproduce	Expected	Actual	Owner	Status
1	Critical	Dashboard doesn't load	1. Fill form 2. Click submit	Show results	Blank screen	FE-2	Open
2	High	OCR timeout	1. Upload large PDF	Extract labs	Timeout error	BE-2	In Progress

3. Performance Testing

Measure:

- □ Form submission → Results: <3 seconds
- □ OCR extraction: <10 seconds
- □ Page load times: <2 seconds
- □ Mobile responsive (all screen sizes)

4. Accessibility Testing

- □ Keyboard navigation works
- □ Screen reader compatible (alt text on images)
- □ Color contrast sufficient (WCAG AA)
- □ Form labels present

Timeline:

- **Day 3 (28th):** Start testing as features complete
- **Day 4 (29th):** Full regression testing
- **Day 5 (30th):** ✅ All critical bugs fixed
- **Day 6 (31st):** Final smoke test before submission

📅 DAILY MILESTONE CHECKLIST

Day 1 (26th) - Architecture & Setup ✅ COMPLETED

All Teams:

- ☑ Repo created and cloned
- ☑ Tech stack decided
- ☑ API contracts frozen
- ☑ Project structure agreed

Frontend:

☑ React app initialized
☑ Tailwind configured
☑ Component structure planned

Backend:

☑ FastAPI skeleton created
☑ Health endpoint working

AI:

☑ AI-2 module complete (all files)
☑ AI-1 OCR strategy decided

Presentation:

☑ Slide outline created

Day 2 (27th) - Core Build I 🔥 TODAY

Frontend:

□ FE-1: All 4 forms UI complete with validation
□ FE-1: Forms send correct JSON format to backend
□ FE-2: Dashboard skeleton with dummy data
□ FE-3: Upload component basic UI

Backend:

□ BE-1: Backend running on localhost:8000
□ BE-1: CORS configured, frontend can call backend
□ BE-2: `/ocr` endpoint skeleton (accepts file)
□ BE-3: `/predict-risk` endpoint returns dummy JSON

AI:

□ AI-1: OCR extracts text from sample image
□ AI-2: Test `predict_risks()` with demo case

Integration:

□ API Lead: Contract doc shared with team
□ API Lead: Postman collection created
□ Frontend can call `/predict-risk` (even with dummy response)

Presentation:

□ Problem statement slides drafted

□ Screenshot collection started

Day 3 (28th) - Core Build II + Integration

Frontend:

□ FE-2: Dashboard displays real API results

□ FE-3: Upload working, calls `/ocr` endpoint

□ FE-3: Extracted labs displayed to user

Backend:

□ BE-2: `/ocr` returns structured lab dict

□ BE-3: `/predict-risk` calls AI-2, returns real predictions

□ All endpoints stable

AI:

□ AI-1: Parser extracts HbA1c, glucose, lipids correctly

□ AI-1: Unit normalizer working

□ AI-2: Predictions work with OCR-extracted labs

Integration:

□ **FULL FLOW WORKS:**

- Upload report → OCR extracts labs → Predict → Dashboard shows results

□ API Lead: All endpoints tested in Postman

Presentation:

□ Solution overview slides done

□ Demo cases prepared

Day 4 (29th) - Detail Pages + Coaching Features

Frontend:

□ FE-3: Disease detail page complete

□ FE-3: All sections rendering (reasons, prevention, tests, consult)

□ Navigation between pages smooth

□ Error states handled

Backend:

- □ Error handling on all endpoints
- □ Validation with Pydantic models
- □ Performance optimized

AI:

- □ AI-1: Test with multiple report formats
- □ AI-2: Fine-tune prevention recommendations
- □ Edge cases handled (missing data)

Presentation:

- □ PPT 60% complete
- □ Screenshots collected from prototype
- □ Demo script written

Day 5 (30th) - Polish + Demo Prep

Frontend:

- □ UI polish (colors, spacing, fonts)
- □ Mobile responsive
- □ Loading states everywhere
- □ Success/error toast messages

Backend:

- □ Deployment attempted (optional)
- □ All endpoints documented

AI:

- □ All bugs fixed
- □ Code commented

Presentation:

- □ PPT 100% complete
- □ Demo video recorded (2-3 min)
- □ Demo rehearsed 2x

All:

- □ No critical bugs
- □ Integration stable

Day 6 (31st) - Final Demo + Submission

All Teams:

- □ Final demo rehearsal (full team)
- □ Reflection video recorded
- □ GitHub repo cleaned up
- □ README updated

Submission Package:

- □ PPT/PDF exported
- □ Demo video uploaded
- □ Reflection video uploaded
- □ GitHub link verified
- □ Live prototype link (if deployed)
- □ All links working

Final Checks:

- □ Can run full demo without errors
- □ Backup plan ready (screenshots if live demo fails)
- □ Q&A prep done

🚨 CURRENT PRIORITY TASKS (Day 2 - January 27th)

What Each Person Should Do RIGHT NOW:

Person	Task	Deadline	Priority
FE-1	Build all 4 forms (basic info, lifestyle, family history) with validation. Must send correct JSON to backend by EOD.	Today 6pm	🔴 Critical
FE-2	Build dashboard skeleton with hardcoded data. Style table with risk class colors.	Today 6pm	🔴 Critical
FE-3	Build upload component UI. Add file picker and preview.	Today 6pm	🟡 High
BE-1	Get FastAPI running with CORS. Test that frontend can call it. Create `/health` endpoint.	Today 4pm	🔴 Critical
BE-2	Create `/ocr` endpoint that accepts file upload. Return dummy JSON for now.	Today 6pm	🟡 High

Person	Task	Deadline	Priority
BE-3	Create `/predict-risk` endpoint. Import AI-2's `predict_risks()`. Test with demo case.	Today 6pm	🔴 Critical
API Lead	Share API contract doc with team. Create Postman collection with all 3 endpoints. Test `/health` endpoint.	Today 5pm	🔴 Critical
AI-1	Get EasyOCR to extract text from one sample medical report image. Print extracted text.	Today 6pm	🔴 Critical
AI-2	Test `predict_risks()` function works with demo case. Help BE-3 integrate it.	Today 4pm	🔴 Critical
PPT	Finish problem statement slides (2-3 slides). Start collecting prototype screenshots.	Today 7pm	🟢 Medium

---

💬 COMMUNICATION PROTOCOL

Daily Standup (15 minutes - Every morning 10am)

Each person answers:

1. **What I completed yesterday**
2. **What I'm working on today**
3. **Any blockers or help needed**

Format: Quick round-robin, no discussions during standup

Shared Resources

1. API Contract Document (Google Doc)

- Owner: API Lead
- All endpoint specs
- Request/response examples
- Updated daily

2. Bug List (Google Sheets)

- Owner: QA / API Lead
- Priority levels: Critical, High, Medium, Low

- Status: Open, In Progress, Fixed

3. Daily Progress Log (Slack Channel)

- End of day: Each person posts what they completed
- Screenshots encouraged
- Blockers flagged with @mentions

4. GitHub Repository

- Branching strategy: `feature/frontend-forms`, `feature/backend-ocr`, etc.
- Pull requests reviewed by 1 other person
- Merge to `main` only when feature complete

Communication Channels

Slack/Discord Channels:

- `#general` - Announcements, standups
- `#frontend` - FE team coordination
- `#backend` - BE team coordination
- `#ai` - AI team coordination
- `#integration` - Cross-team issues
- `#bugs` - Bug reports

Quick Questions:

- DM for quick clarifications
- @mention in channel for urgent blockers

Meetings:

- Daily standup: 15 min
- Integration sync (Days 3, 4): 30 min
- Demo rehearsal (Day 5, 6): 45 min

Escalation Process

If blocked:

1. Try to solve independently (15 min)
2. Ask in relevant Slack channel
3. @mention specific person who can help
4. If critical and unresolved: Call emergency team sync

If behind schedule:

- Notify team immediately
- Assess if feature can be cut or simplified
- Redistribute work if needed

✅ SUCCESS CRITERIA

By Day 6, we must have:

1. ✅ Working prototype with full end-to-end flow
2. ✅ At least ONE demo case working flawlessly (Case B recommended)
3. ✅ Clean, professional UI
4. ✅ Complete presentation materials (PPT + videos)
5. ✅ GitHub repo with README
6. ✅ Rehearsed demo (3 min max)

Nice to have:

- Deployed version (Vercel/Railway)
- Multiple demo cases
- Charts/visualizations
- Mobile app version

🎯 FINAL NOTES

Remember:

- **Scope creep is the enemy** - Stick to MVP features
- **Integration early and often** - Don't wait until Day 5
- **Working demo > Perfect code** - Prioritize functionality over polish
- **One person, one task at a time** - Avoid context switching

- **Communicate blockers immediately** - Don't suffer in silence

Cut if behind schedule:

1. Database (use in-memory storage)
2. Extra diseases (keep top 5)
3. Charts/visualizations
4. Deployment
5. Advanced form validation

DO NOT CUT:

1. End-to-end flow (Form → Predict → Results)
2. OCR demo (even with 1 sample report)
3. Disease detail page
4. Risk class display
5. At least 1 perfect demo case

Good luck team! Let's build something amazing! 🚀

Document Version: 1.0
Last Updated: January 27, 2026
Next Review: End of Day 2