

# StudyNet AI Agent System Design

## Team Responsibilities

### Kabid (Backend) -

- **Repo:** [ai-agent-backend](#)
- **Main file:** [main.py](#) (already done!)
- **Handle:** All APIs, PDF storage, FAISS, AI responses

### Rayhan (Frontend) -

- **Repo:** [ai-agent-frontend](#)
  - **Build:** HTML pages that call your APIs
  - **Handle:** User interface, file uploads, chat display
- 

## API Endpoints

### 1. System Status APIs

GET / # Health check  
GET /status # RAG system status  
GET /llm-options # Available AI models  
POST /configure-llm # Change AI model

### 2. Document APIs

POST /upload-pdf # Upload new PDF (main upload endpoint!)

### 3. Chat APIs

POST /ask # Ask question to AI (main chat endpoint!)

---

# Simple Folder Structure

## Backend (Kabid)

```
ai-agent-backend/
├── main.py          # Your existing FastAPI app
├── pdfs/            # PDF storage folder
│   ├── existing_pdf1.pdf # Your current PDFs
│   ├── existing_pdf2.pdf
│   └── uploaded_file.pdf # New uploads go here
├── requirements.txt  # Python dependencies
└── README.md
```

## Frontend (Rayhan)

```
ai-agent-frontend/
├── pages/
│   ├── index.html    # Main chat page
│   ├── upload.html   # PDF upload page
│   └── settings.html  # AI model settings
├── css/
│   └── style.css      # All styling
├── js/
│   ├── chat.js        # Chat functionality
│   ├── upload.js       # File upload logic
│   ├── settings.js     # Model configuration
│   └── api.js          # API calls to your backend
└── README.md
```

---

# Simple Workflow

## 1. PDF Upload Flow

```
[Rayhan's upload.html]
  ↓ User selects PDF
[JavaScript FormData]
  ↓ POST to your API
[Your /upload-pdf endpoint]
```

↓ Save to pdfs/ folder  
[Your FAISS system updates]  
↓ Return success  
[Rayhan shows "Upload Complete!"]

## 2. Chat Flow

[Rayhan's index.html]  
↓ User types question  
[JavaScript fetch()]  
↓ POST to your API  
[Your /ask endpoint]  
↓ Search FAISS + AI response  
[Return answer]  
↓ Display in chat  
[Rayhan shows AI response]

---

# Rayhan's Frontend Pages

## 1. Chat Page (index.html)

```
<!DOCTYPE html>
<html>
<head>
  <title>AI Assistant</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <div class="chat-container">
    <div id="chat-messages"></div>
    <div class="input-container">
      <input type="text" id="question-input" placeholder="Ask a
question...">
      <button onclick="askQuestion()">Send</button>
    </div>
  </div>
  <script src="js/api.js"></script>
  <script src="js/chat.js"></script>
```

```
</body>
</html>
```

## 2. Upload Page (upload.html)

```
<!DOCTYPE html>
<html>
<head>
  <title>Upload PDF</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <div class="upload-container">
    <h2>Upload PDF Document</h2>
    <input type="file" id="pdf-file" accept=".pdf">
    <button onclick="uploadPDF()">Upload</button>
    <div id="upload-status"></div>
  </div>
  <script src="js/api.js"></script>
  <script src="js/upload.js"></script>
</body>
</html>
```

## 3. Settings Page (settings.html)

```
<!DOCTYPE html>
<html>
<head>
  <title>AI Settings</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <div class="settings-container">
    <h2>AI Model Settings</h2>
    <select id="llm-provider">
```

```
        <option value="openai">OpenAI</option>
        <option value="groq">Groq</option>
        <option value="gemini">Gemini</option>
    </select>
    <button onclick="configureLLM()">Save Settings</button>
</div>
<script src="js/api.js"></script>
<script src="js/settings.js"></script>
</body>
</html>
```

---

## API Calls (Rayhan's JavaScript)

### API Helper (api.js)

```
const API_BASE = 'http://localhost:8000'; // Your backend URL

// Generic API call function
async function apiCall(endpoint, method = 'GET', data = null) {
    const options = {
        method: method,
        headers: {}
    };

    if (data && !(data instanceof FormData)) {
        options.headers['Content-Type'] = 'application/json';
        options.body = JSON.stringify(data);
    } else if (data) {
        options.body = data;
    }

    const response = await fetch(API_BASE + endpoint, options);
    return response.json();
}
```

## Chat Functions (chat.js)

```
async function askQuestion() {
  const question = document.getElementById('question-input').value;

  const response = await apiCall('/ask', 'POST', {
    question: question,
    llm_provider: 'openai' // or get from settings
  });

  displayMessage(question, 'user');
  displayMessage(response.answer, 'ai');
}

function displayMessage(message, sender) {
  const chatDiv = document.getElementById('chat-messages');
  const messageDiv = document.createElement('div');
  messageDiv.className = sender;
  messageDiv.textContent = message;
  chatDiv.appendChild(messageDiv);
}
```

## Upload Functions (upload.js)

```
async function uploadPDF() {
  const fileInput = document.getElementById('pdf-file');
  const file = fileInput.files[0];

  if (!file) {
    alert('Please select a PDF file');
    return;
  }

  const formData = new FormData();
  formData.append('file', file);

  const response = await fetch(API_BASE + '/upload-pdf', {
    method: 'POST',
```

```
    body: formData
  });


  const result = await response.json();
  document.getElementById('upload-status').textContent = result.message;
}
```

---

## Development Plan (2 Weeks)

### : Basic Connection

#### Kabid (You):

-  Your main.py is already done!
- Just run: `uvicorn main:app --reload`
- Test your APIs with curl or Postman

#### Rayhan:

- Create basic HTML files
- Test connection to your `/status` endpoint
- Build simple chat interface

### Week 2: Full Integration

#### Kabid (You):

- Test file uploads thoroughly
- Add any missing error handling
- Help debug API connection issues

#### Rayhan:

- Complete file upload interface
  - Style the pages with CSS
  - Test end-to-end functionality
-

# How They Connect

## Your Backend Runs On:

```
cd ai-agent-backend
uvicorn main:app --reload --host 0.0.0.0 --port 8000
# Available at: http://YOUR_IP:8000
```

## His Frontend Calls:

```
// He updates this to your actual IP
const API_BASE = 'http://YOUR_MACHINE_IP:8000';
```

---

## Success Criteria

- ✓ **Rayhan can upload PDF** → File appears in your **pdfs/** folder
- ✓ **Rayhan can ask questions** → Your AI responds using uploaded PDFs
- ✓ **Clean, simple interface** → Easy to use and understand
- ✓ **Error handling** → Shows helpful messages when things go wrong

## Next Steps

1. **You:** Share your machine's IP address with Rayhan
2. **Rayhan:** Update **API\_BASE** in his JavaScript files
3. **Both:** Test file upload → chat workflow
4. **Rayhan:** Add styling and improve user experience

Your backend is already solid! Now it's just about building a simple frontend that talks to your existing APIs. 🚀



