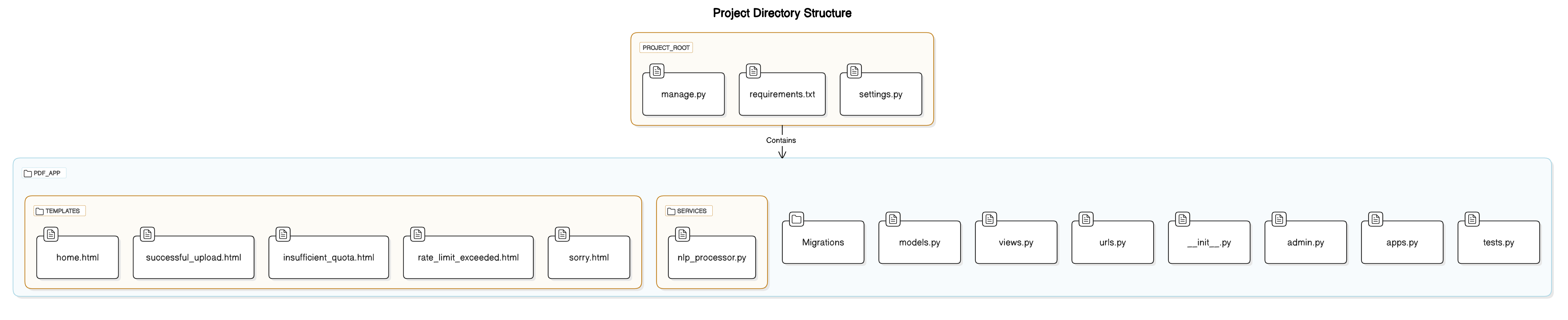
# PDF Question-Answering Application - Source Code Documentation

## 1. Project Structure



## 2. Key Components

### 2.1 Models

Document Model:  
- Stores details about uploaded documents (filename, extracted text content).  
- Fields: filename, content, uploaded\_at

### 2.2 Views

home: Renders main page (home.html) for PDF upload.  
  
upload\_pdf: Handles file upload, text extraction, and storage.  
- Saves extracted text in database and session; redirects to successful upload page.  
  
successful\_upload: Displays extracted text post-upload.  
  
ask\_question: Retrieves answer based on user question and document text.  
- Handles errors by redirecting users to relevant pages (e.g., insufficient quota).  
  
insufficient\_quota, rate\_limit\_exceeded, sorry: Render specific error templates.

### 2.3 Text Extraction Service

extract\_text\_from\_pdf(file\_path): Extracts text from PDF pages using PyMuPDF.

### 2.4 NLP Processor

get\_answer: Processes question using an NLP API, raises custom errors for quota and rate limit issues.

## 3. Interactions and Data Flow

1. Home Page Access: User accesses home page for PDF upload.  
  
2. PDF Upload: upload\_pdf saves the file, extracts text, and stores it in database and session.  
  
3. Question Submission: ask\_question sends document text and question to NLP processor for an answer.  
  
4. Error Handling: Specific errors redirect users to appropriate pages.

## 4. URL Routing

URLs mapped to views in urls.py:  
- home, upload\_pdf, successful\_upload, ask\_question, insufficient\_quota, rate\_limit\_exceeded, sorry.