

TalentScout - AI-Powered Hiring Assistant

Complete Documentation Package

TABLE OF CONTENTS

1. [Project Overview](#)
 2. [Features & Capabilities](#)
 3. [Complete Setup Guide](#)
 4. [Architecture & Design](#)
 5. [User Guide](#)
 6. [Developer Guide](#)
 7. [Troubleshooting](#)
 8. [FAQ](#)
-






PROJECT OVERVIEW

What is TalentScout?

TalentScout is an intelligent, AI-powered hiring assistant designed to streamline the initial candidate screening process for technology recruitment. Built with Streamlit and powered by Groq's LLaMA 3.3 70B model, it automates the time-consuming task of conducting first-round technical interviews.





Why TalentScout?

Traditional Hiring Challenges:

-  Manual screening takes 20-30 minutes per candidate
-  Inconsistent interview questions across candidates
-  No standardized evaluation criteria
-  Time-consuming report generation
-  Difficult to track and compare candidates

TalentScout Solutions:

-  Automated screening in 10-15 minutes

-  Consistent, tech-stack-specific questions
-  AI-powered analysis with scoring
-  Instant PDF and JSON reports
-  Organized candidate database

Key Statistics

- **Time Saved:** 50% reduction in initial screening time
 - **Consistency:** 100% standardized evaluation process
 - **Scalability:** Screen unlimited candidates simultaneously
 - **Accuracy:** AI-powered analysis with context awareness
 - **Organization:** Automatic report generation and filing
-

FEATURES & CAPABILITIES

1. Dual Input Modes

Chat Mode

Perfect for: Interactive interviews, candidates without resumes

- Real-time conversational interface
- One question at a time approach
- Natural language processing
- Context-aware follow-ups

Resume Upload Mode

Perfect for: Bulk screening, pre-qualified candidates






- PDF resume parsing
- Automatic information extraction
- Smart validation of extracted data
- Only asks for missing information

2. Intelligent Information Gathering

Required Information (7 Fields):

1. **Full Name** - Candidate identification
2. **Email Address** - Contact information
3. **Phone Number** - Alternative contact
4. **Years of Experience** - Seniority assessment
5. **Desired Position(s)** - Role alignment
6. **Current Location** - Geographical fit
7. **Tech Stack** - Technical expertise inventory

Smart Features:

-  Skips already-known information
-  Validates data format (email, phone)
-  Natural language conversation flow
-  No repetitive questions
-  Context-aware prompting

3. Technical Assessment Engine

Question Generation:

- **Personalized:** Based on candidate's tech stack
- **Adaptive:** Matches experience level
- **Scenario-Based:** Practical problem-solving
- **Consistent:** Exactly 5 questions per candidate
- **Comprehensive:** Covers breadth and depth

Example Questions by Technology:

Python Developer:

- "Explain the difference between lists and tuples with use cases"
- "How do you handle memory management in Python applications?"
- "Describe your approach to writing unit tests in Python"

React Developer:

- "What's your strategy for state management in large applications?"

- "Explain the difference between class and functional components"
- "How do you optimize React app performance?"

DevOps Engineer:

- "Describe your experience with CI/CD pipelines"
- "How do you approach container orchestration?"
- "What monitoring tools have you used and why?"






4. Voice Input Integration

Powered by Google Speech Recognition

Activation:

- Automatically enabled during technical Q&A phase
- Toggle on/off in sidebar
- Fallback to text input anytime

Features:

-  Real-time voice recording (15 seconds)
-  Automatic speech-to-text transcription
-  Instant transcription preview
-  Re-record option if transcription fails
-  Seamless text input alternative

Use Cases:

- Hands-free answering for accessibility
- Natural conversation flow
- Candidates who prefer speaking
- Testing communication skills

5. AI-Powered Analysis

Comprehensive Evaluation:

Technical Competency Score (0-10)

- Overall technical knowledge rating

- Based on answer quality and depth

Strengths Analysis

- Specific examples from answers
- Technical expertise highlights
- Communication effectiveness

Areas for Improvement

- Knowledge gaps identified
- Constructive feedback
- Learning recommendations

Knowledge Depth Assessment

- Beginner / Intermediate / Advanced / Expert
- Technology-specific evaluation
- Experience alignment verification

Communication Skills





- Clarity of explanations
- Technical vocabulary usage
- Problem-solving approach



Hiring Recommendation

- Strong Hire / Hire / Maybe / No Hire
- Detailed reasoning
- Suggested next steps






6. Professional Report Generation

PDF Report Features:

-  Professional formatting with colors
-  Structured information tables
-  Complete Q&A session
-  AI analysis with scoring

-  Multi-page support with headers
-  Brand-consistent design

JSON Report Features:

-  Structured data format
-  Easy database integration
-  Searchable content
-  API-ready format
-  Analytics-friendly

Report Contents:

CANDIDATE REPORT

- └─ Header (Name, Date, Time)
- └─ Candidate Information Table
 - └─ Contact Details
 - └─ Experience Level
 - └─ Desired Roles
 - └─ Tech Stack
- └─ Technical Assessment
 - └─ Question 1 + Answer
 - └─ Question 2 + Answer
 - └─ Question 3 + Answer
 - └─ Question 4 + Answer
 - └─ Question 5 + Answer
- └─ AI Analysis
 - └─ Competency Score
 - └─ Strengths
 - └─ Improvements
 - └─ Knowledge Depth

└— Communication

└— Recommendation

File Naming Convention:

Format: {CandidateName}_{YYYYMMDD}_{HHMMSS}.{extension}

Examples:

✓ John_Doe_20241119_143022.pdf

✓ Jane_Smith_20241119_150815.json

✓ Robert_Johnson_20241120_091530.pdf

Storage: /Reports/{filename}

7. Automatic Workflow

Session Flow:

1. **Start** → Mode selection
2. **Collect** → Gather 7 required fields
3. **Assess** → Ask 5 technical questions
4. **Analyze** → AI generates evaluation
5. **Report** → Create PDF + JSON
6. **Complete** → Auto-reset for next candidate

Auto-Detection:

- ✓ Knows when all info is collected
 - ✓ Transitions to technical phase
 - ✓ Counts questions (stops at 5)
 - ✓ Detects completion phrases
 - ✓ Triggers report generation
 - ✓ Resets for new session
-

COMPLETE SETUP GUIDE

Prerequisites

System Requirements:

- **Operating System:** Windows 10/11, macOS 10.14+, Linux (Ubuntu 18.04+)
- **Python:** Version 3.8, 3.9, 3.10, or 3.11
- **RAM:** Minimum 4GB (8GB recommended)
- **Storage:** 500MB free space
- **Internet:** Required for API calls
- **Microphone:** Optional (for voice input)

Required Accounts:

- **Groq API Account:** [Sign up here](#)
 - Free tier available
 - API key required

Step-by-Step Installation

1 Install Python

Windows:

Download from python.org

Or use winget

winget install Python.Python.3.11

Verify installation

python --version

macOS:

Using Homebrew

brew install python@3.11

Verify installation

```
python3 --version
```

Linux (Ubuntu/Debian):

```
sudo apt update
```

```
sudo apt install python3.11 python3-pip python3-venv
```

```
# Verify installation
```

```
python3 --version
```

2 Create Project Directory

```
# Create and navigate to project folder
```

```
mkdir talentscout
```

```
cd talentscout
```

3 Create Virtual Environment (Recommended)

Why Virtual Environment?

- Isolates project dependencies
- Prevents version conflicts
- Easy to reproduce setup

Windows:

```
python -m venv venv
```

```
venv\Scripts\activate
```

macOS/Linux:

```
python3 -m venv venv
```

```
source venv/bin/activate
```

You should see (venv) prefix in your terminal

4 Create All Project Files

Quick Method (Unix/Mac/Linux):

```
touch config.py prompts.py utils.py llm_handler.py \
```

```
voice_handler.py report_generator.py main.py \
```

```
requirements.txt README.md
```

Windows PowerShell:

New-Item config.py, prompts.py, utils.py, llm_handler.py, `
voice_handler.py, report_generator.py, main.py, `
requirements.txt, README.md

Manual Method:

1. Open your code editor (VS Code, PyCharm, etc.)
2. Create each file manually
3. Save in the talentscout folder

5 Copy Code to Files

From the main artifact, copy each section:

config.py

Copy lines 1-60 from artifact

File: config.py

"""Configuration settings for TalentScout application."""

import streamlit as st

from dataclasses import dataclass

...

prompts.py

Copy lines 61-200 from artifact

File: prompts.py

"""System prompts and prompt templates."""

...

utils.py

Copy lines 201-310 from artifact

File: utils.py

"""Utility functions for file processing and data formatting."""

...

llm_handler.py

Copy lines 311-400 from artifact

File: llm_handler.py

"""LLM initialization and chain creation."""

...

voice_handler.py

Copy lines 401-510 from artifact

File: voice_handler.py

"""Voice input handling using Google Speech Recognition."""

...

report_generator.py

Copy lines 511-780 from artifact

File: report_generator.py

"""PDF and JSON report generation."""

...

main.py

Copy lines 781-end from artifact

File: main.py (app.py)

"""Main Streamlit application."""

...

6 Create requirements.txt

streamlit>=1.28.0

langchain-groq>=0.1.0

langchain-core>=0.1.0

pdfplumber>=0.10.0

reportlab>=4.0.0

SpeechRecognition>=3.10.0

PyAudio>=0.2.13

7 Install Python Dependencies

```
pip install --upgrade pip
```

```
pip install -r requirements.txt
```

This installs:

- Streamlit (Web framework)
- LangChain-Groq (AI integration)
- PDFPlumber (PDF parsing)
- ReportLab (PDF generation)
- SpeechRecognition (Voice input)
- PyAudio (Audio recording)

8 Install PyAudio (Special Setup)

PyAudio is platform-specific and requires extra steps

Windows:

```
# Install pipwin (PyAudio helper)
```

```
pip install pipwin
```

```
# Install PyAudio
```

```
pipwin install pyaudio
```

```
# Alternative: Download wheel file
```

```
# Visit: https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyaudio
```

```
# Download matching Python version
```

```
# pip install PyAudio-0.2.11-cp311-cp311-win_amd64.whl
```

macOS:

```
# Install portaudio dependency
```

```
brew install portaudio
```

```
# Install PyAudio
```

```
pip install pyaudio
```

Linux (Ubuntu/Debian):

Install dependencies

```
sudo apt-get update
```

```
sudo apt-get install portaudio19-dev python3-pyaudio
```

Install PyAudio

```
pip install pyaudio
```

Verification:

```
python -c "import pyaudio; print('PyAudio installed successfully')"
```

Get Groq API Key

1. Visit <https://console.groq.com>
2. Sign up / Log in
3. Navigate to **API Keys** section
4. Click **Create API Key**
5. Copy the key (starts with gsk_...)
6. **Keep it secure!** Don't commit to Git

Verify Installation

Check Python modules:

```
python -c "import streamlit; print('✅ Streamlit')"
```

```
python -c "import langchain_groq; print('✅ LangChain-Groq')"
```

```
python -c "import pdfplumber; print('✅ PDFPlumber')"
```

```
python -c "import reportlab; print('✅ ReportLab')"
```

```
python -c "import speech_recognition; print('✅ SpeechRecognition')"
```

```
python -c "import pyaudio; print('✅ PyAudio')"
```

Check file structure:

```
ls -la
```

Should show all 9 files

Test imports:

```
python -c "import config, prompts, utils, llm_handler, voice_handler, report_generator"
```

First Run

1. Start the Application

```
streamlit run main.py
```

Expected Output:

You can now view your Streamlit app in your browser.



Local URL: `http://localhost:8501`

Network URL: `http://192.168.1.x:8501`

2. Open Browser


- Automatically opens at `http://localhost:8501`
- Or manually navigate to the URL

3. Enter API Key

- Look for sidebar on the left
- Find " Configuration" section
- Paste your Groq API key
- Click outside the input field
- See " API Key configured" message

4. Test with Sample Screening

Quick Test Flow:

1. Click " Start Chat"
2. Enter name: "John Doe"
3. Enter email: "john@example.com"
4. Enter phone: "+1-555-0123"
5. Enter experience: "5"
6. Enter position: "Senior Python Developer"

7. Enter location: "New York, NY"
8. Enter tech stack: "Python, Django, PostgreSQL, AWS"
9. Answer 5 technical questions
10. View generated reports

5. Verify Report Generation

Check Reports/ folder:

ls -la Reports/

Should show PDF and JSON files

Configuration Options

Customize Application

Edit config.py:

```
@dataclass
```

```
class AppConfig:
```

```
    MODEL_NAME: str = "llama-3.3-70b-versatile" # Change AI model
```

```
    TEMPERATURE: float = 0 # Creativity (0-1)
```

```
    MAX_RETRIES: int = 2 # API retry attempts
```

```
    REPORTS_FOLDER: str = "Reports" # Change folder name
```

Modify Question Count

Edit prompts.py (line ~50):

```
# Change from 5 to desired number
```

```
- Generate exactly 5 technical questions based on their tech stack
```

Change Voice Recording Duration

Edit voice_handler.py:

```
def get_voice_input(duration: int = 15): # Change 15 to desired seconds
```

Customize Report Styling

Edit report_generator.py:

```
title_style = ParagraphStyle(
```

```
'CustomTitle',  
  
fontSize=24, # Change size  
  
textColor=colors.HexColor('#1E88E5'), # Change color  
  
...  
)
```

Security Best Practices

API Key Management:

DO:

- Store in environment variables
- Use .env file (not committed to Git)
- Rotate keys regularly
- Use different keys for dev/prod

DON'T:

- Hardcode in source files
- Commit to version control
- Share in screenshots
- Use in public repositories

Using .env file:

1. Install python-dotenv:

```
pip install python-dotenv
```

2. Create .env file:

```
GROQ_API_KEY=gsk_your_key_here
```

3. Modify main.py:

```
from dotenv import load_dotenv
```

```
import os
```

```
load_dotenv()
```

```
api_key = os.getenv("GROQ_API_KEY")
```

4. Add to .gitignore:

```
.env
```

Final Directory Structure

talentscout/


├─ venv/ # Virtual environment (don't commit)


├─ config.py #  Created

├─ prompts.py #  Created

├─ utils.py #  Created

├─ llm_handler.py #  Created

├─ voice_handler.py #  Created

├─ report_generator.py #  Created

├─ main.py #  Created

├─ requirements.txt #  Created

├─ README.md #  Created

├─ .env # Optional (API key)

├─ .gitignore # Optional (Git)

└─ Reports/ # Auto-created on first run

├─ John_Doe_20241119_143022.pdf

└─ John_Doe_20241119_143022.json

USER GUIDE

For Recruiters

Starting a Screening Session

1. Launch Application

2. streamlit run main.py

3. Choose Input Mode

- **Chat Mode:** For live interviews or phone screens
- **Resume Mode:** For pre-screened candidates with resumes

4. Information Collection Phase

- AI asks for 7 required fields
- Provide accurate information
- Correct any mistakes immediately

5. Technical Assessment Phase





- AI asks 5 personalized questions
- Enable voice input for natural conversation
- Take notes if needed

6. Review and Download





- AI generates comprehensive report
- Download PDF for presentation
- Download JSON for database storage
- Compare with other candidates

Best Practices for Recruiters

Before Screening:

-  Prepare candidate's resume (if available)
-  Check audio equipment for voice mode
-  Review tech stack requirements
-  Allocate 15-20 minutes per candidate

During Screening:

-  Let AI lead the conversation
-  Don't interrupt the flow
-  Encourage complete answers
-  Use voice input for efficiency

After Screening:

- ☒ Review AI analysis carefully
- ☒ Compare scores across candidates
- ☒ Use report in hiring decisions
- ☒ Store reports systematically

For Candidates

What to Expect

Phase 1: Information Collection (5-7 minutes)

- Personal details
- Experience level
- Career goals
- Technical skills

Phase 2: Technical Assessment (10-15 minutes)

- 5 technical questions
- Based on your tech stack
- Scenario-based problems
- Communication evaluation

Tips for Success

Preparation:





- ☒ Have resume ready
- ☒ Test microphone (if using voice)
- ☒ Review your tech stack
- ☒ Prepare examples

During Interview:

- ☒ Be specific and detailed
- ☒ Explain your thinking process
- ☒ Use technical terminology correctly

-  Ask for clarification if needed

Communication:

-  Speak clearly (voice mode)
 -  Structure your answers
 -  Provide real-world examples
 -  Be honest about experience
-



DEVELOPER GUIDE

Architecture Overview

Module Responsibilities

config.py - Configuration Layer

- Application settings
- Session state management
- Global constants

prompts.py - Prompt Engineering Layer

- System prompts
- Template generation
- Context formatting

utils.py - Utility Layer

- File processing
- Data formatting
- Helper functions

llm_handler.py - AI Integration Layer

- LLM initialization
- Chain management
- AI operations

voice_handler.py - Voice Input Layer

- Speech recognition
- Audio recording
- Transcription

report_generator.py - Report Generation Layer

- PDF creation
- JSON formatting
- File management

main.py - Application Layer

- UI components
- User interaction
- Workflow orchestration

Data Flow

User Input

↓

main.py (UI)

↓

llm_handler.py (Process with AI)

↓

prompts.py (Generate prompts)

↓

Groq API (LLaMA 3.3 70B)

↓

llm_handler.py (Parse response)

↓

main.py (Update UI)

↓

report_generator.py (Create reports)

↓

Reports/ folder

Extending Functionality

Add New Required Field

1. Update config.py:

```
REQUIRED_FIELDS = [  
    "full_name",  
    "email",  
    "phone_number",  
    "years_of_experience",  
    "desired_positions",  
    "current_location",  
    "tech_stack",  
    "linkedin_profile" # NEW FIELD  
]
```

2. Update prompts.py:

REQUIRED INFORMATION TO COLLECT:

1. Full Name

2. Email Address

...

8. LinkedIn Profile # NEW FIELD

3. Update extraction prompt:

Required fields:

- full_name (string)

...

- linkedin_profile (string) # NEW FIELD

Add Custom Analysis Criteria

Edit prompts.py - get_analysis_prompt():

Provide a comprehensive analysis covering:

1. Overall Technical Competency
2. Strengths
3. Areas for Improvement
4. Knowledge Depth Assessment
5. Communication Skills
6. Cultural Fit Assessment # NEW CRITERIA
7. Leadership Potential # NEW CRITERIA
8. Recommendation