**Guide of Reality Defender**

## **Introduction**

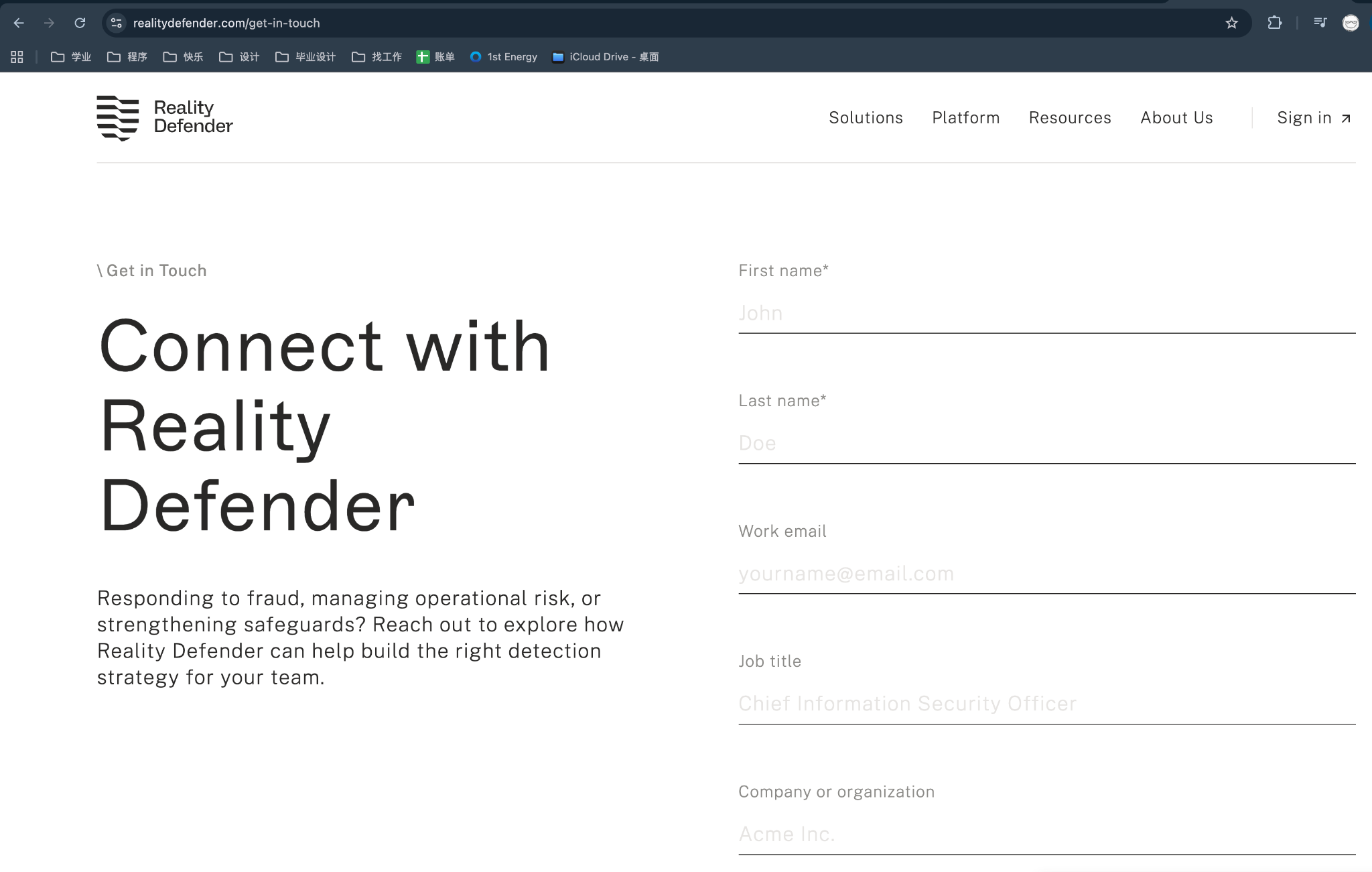
Reality Defender is an AI agent dedicated to detecting deepfake videos, images, audio, and text. It provides SDKs that can be integrated into projects written in Java, Python, TypeScript, and other languages. In September, Reality Defender engineer **Manuel Abeledo** (manuel@realitydefender.ai) built an [SDK](https://github.com/Reality-Defender/realitydefender-sdk-java/tree/manuel/added-optional-deps-profile) based on our project specifically for Android development. After testing, I confirmed that this SDK can be successfully deployed and run.

At the beginning of August, I registered a free account on [Reality Defender’s official website](https://www.realitydefender.com/) and used my university email to present our project requirements to the team. The company was very enthusiastic and provided a sufficient quota of AI detections for our project.

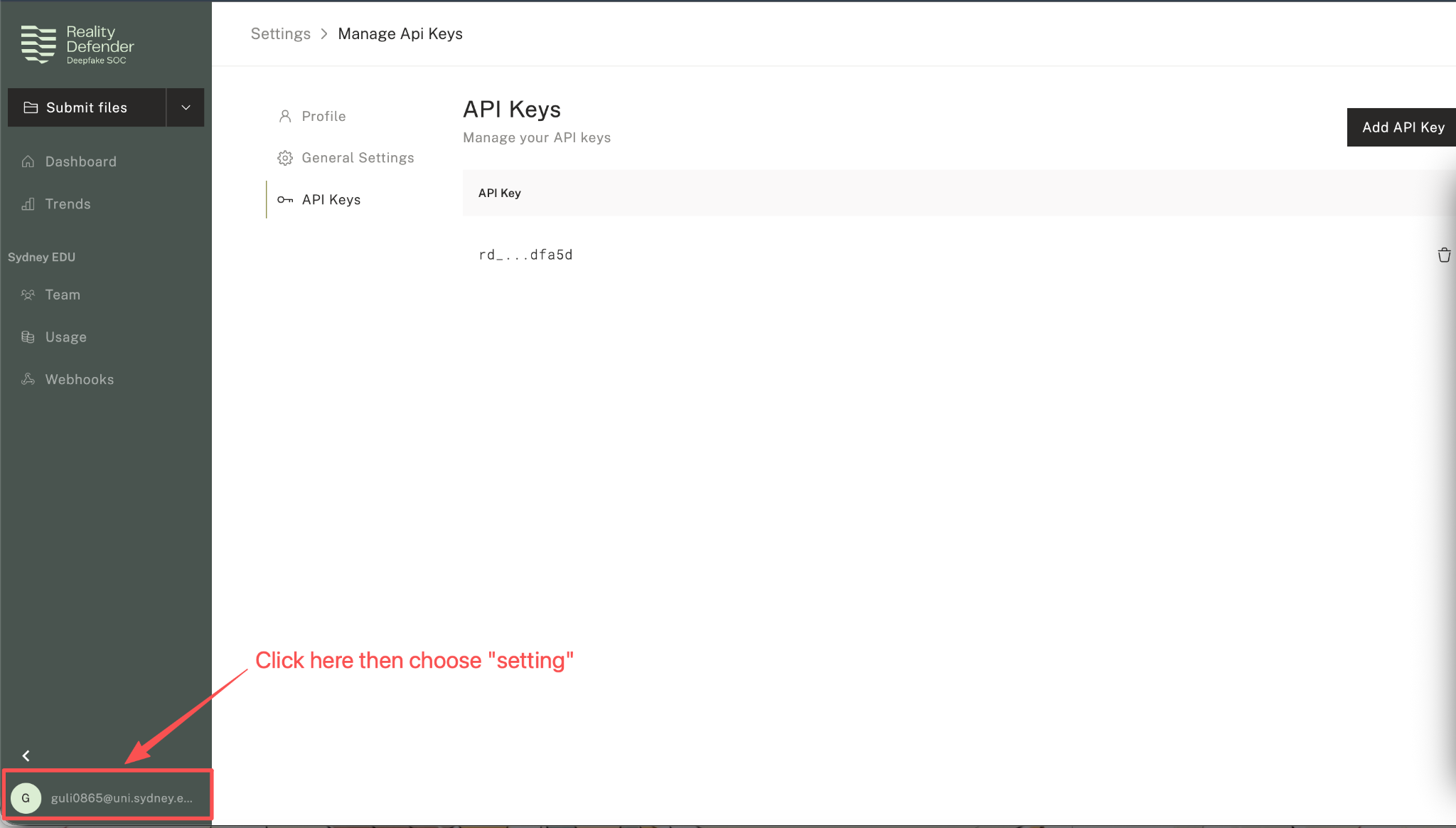
## **Deployment**

1. Set API key

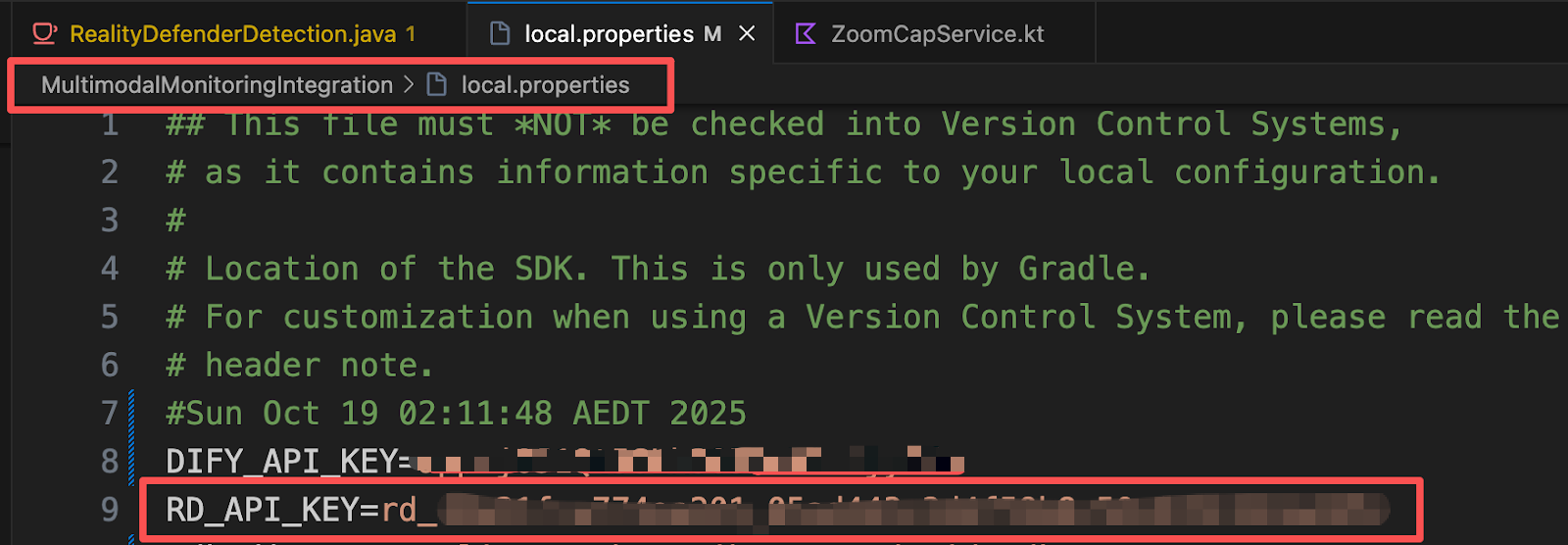
Currently, the RD testing part of this APP still uses my own account and API-key. It has limited testing sessions that are around 20000. Therefore, **I recommend you proactively to contact Reality Defender** — they are eager to collaborate with students and researchers (even better if there will be a published paper). Besides free media detection, Reality Defender also offers real-time detection services (paid). They can create a new account for you with access to more services and provide ample technical support. ([Contact page link](https://www.realitydefender.com/) here, open then click **get in touch**. Below picture shows the get-in-touch page.)



After creating an account, you can add your own **API key** in user settings and replace "REALITY\_DEFENDER\_API\_KEY" in the sample functions to start using them. In addition, each detection—whether for video, image, audio, or text—consumes one detection credit, and every detection is assigned a unique **requestID** that can be used to retrieve historical results.

You can add and find your API keys in your **dashboard**, like the below picture shows:

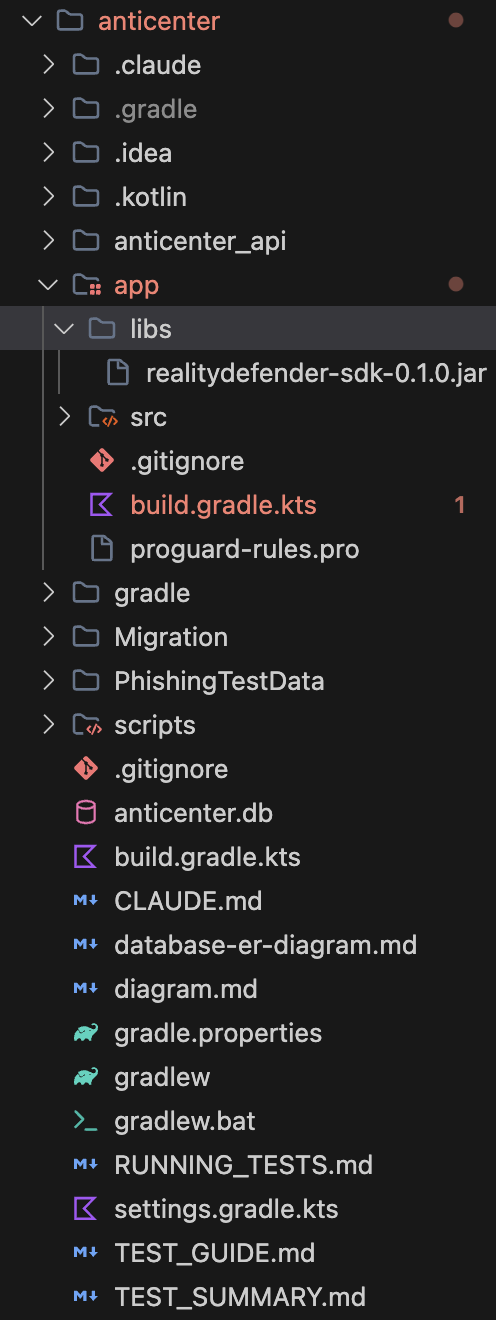
In our project, if you want to change the API key into yours, you need to change it in anticenter/local.properties (when you open the program in Android studio, you can see this file) which shows in RD\_API\_KEY.



1. SDK deployment

Reality Defender provides a white paper for its SDKs along with [GitHub repository links](https://docs.realitydefender.com/sdks/quickstart) for SDKs in various programming languages, and the repositories include detailed deployment instructions for each SDK.

Taking the [**Java SDK**](https://github.com/Reality-Defender/realitydefender-sdk-java/tree/manuel/added-optional-deps-profile) as an example, Reality Defender documents how to configure it for projects managed with **Maven** or **Gradle**. For Gradle, if you use **Groovy** DSL in your project, please add implementation 'ai.realitydefender:realitydefender-sdk:[0.1.0,0.2.0)' to your project’s build.gradle file. If you use **Kotlin** DSL, please add implementation(“ai.realitydefender:realitydefender-sdk:[0.1.0,0.2.0)”) to your project’s build.gradle.kts file.

In our project, we use the [SDK](https://github.com/Reality-Defender/realitydefender-sdk-java/tree/manuel/added-optional-deps-profile) that is compatible with Android development. If you want to deploy this SDK on your own project, you should firstly pull the SDK to the local environment, then use mvn clean package -Pminimal -DskipTests to build a jar. After that, put that jar in anticenter/app/libs which shows in left picture, and add implementation(“ai.realitydefender:realitydefender-sdk:0.1.0”) or implementation 'ai.realitydefender:realitydefender-sdk:0.1.0' to build.gradle.kts or build.gradle file’s dependencies{} part, then rebuild the Gradle.

1. How to detect files and get the detection results

There are some example functions in the [examples folder](https://github.com/Reality-Defender/realitydefender-sdk-java/blob/manuel/added-optional-deps-profile/src/main/java/ai/realitydefender/examples/SimpleFileDetectionExample.java):

* SimpleFileDetectionExample.java — a minimal example that uploads a file and returns the detection result in real time,
* SocialMediaDetectionExample.java — an example that analyzes the content available at a social media URL, and
* GetResultsExample.java — an example that fetches past detection results via a requestID.

If you want to achieve more functions, please make modifications and programming based on your own requirements on the basis of these sample functions. For instance, in our project, I wrote submit, runWithRetry, wrapWithTimeout functions to realise asynchronous uploading of files.