

Integration Document

Flexible Backend for Easy Integration

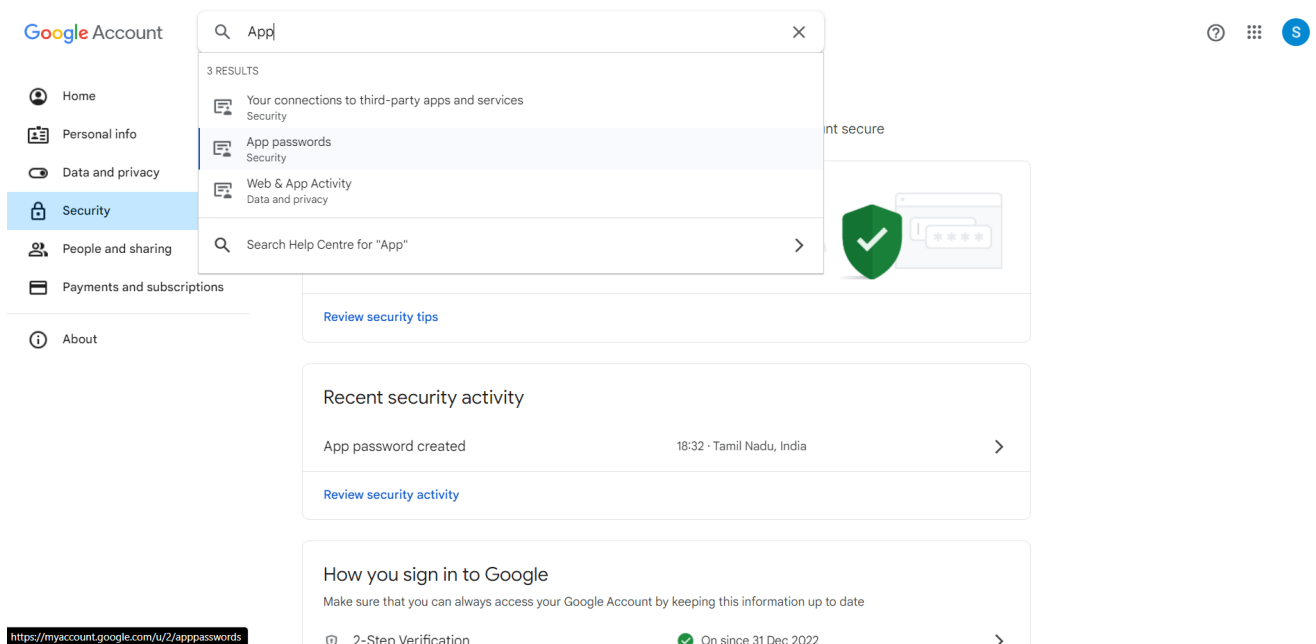
1. **Environment-Friendly Backend:** We've designed our backend to work smoothly in any setup. Whether it's for testing, production, or anything in between, our system doesn't require code changes to adapt.
2. **Easy Third-Party Additions:** Adding external tools and services is a breeze. Our School Technical Support team can set things up without touching the code. It's all about simplicity and flexibility.

Setting up School Name

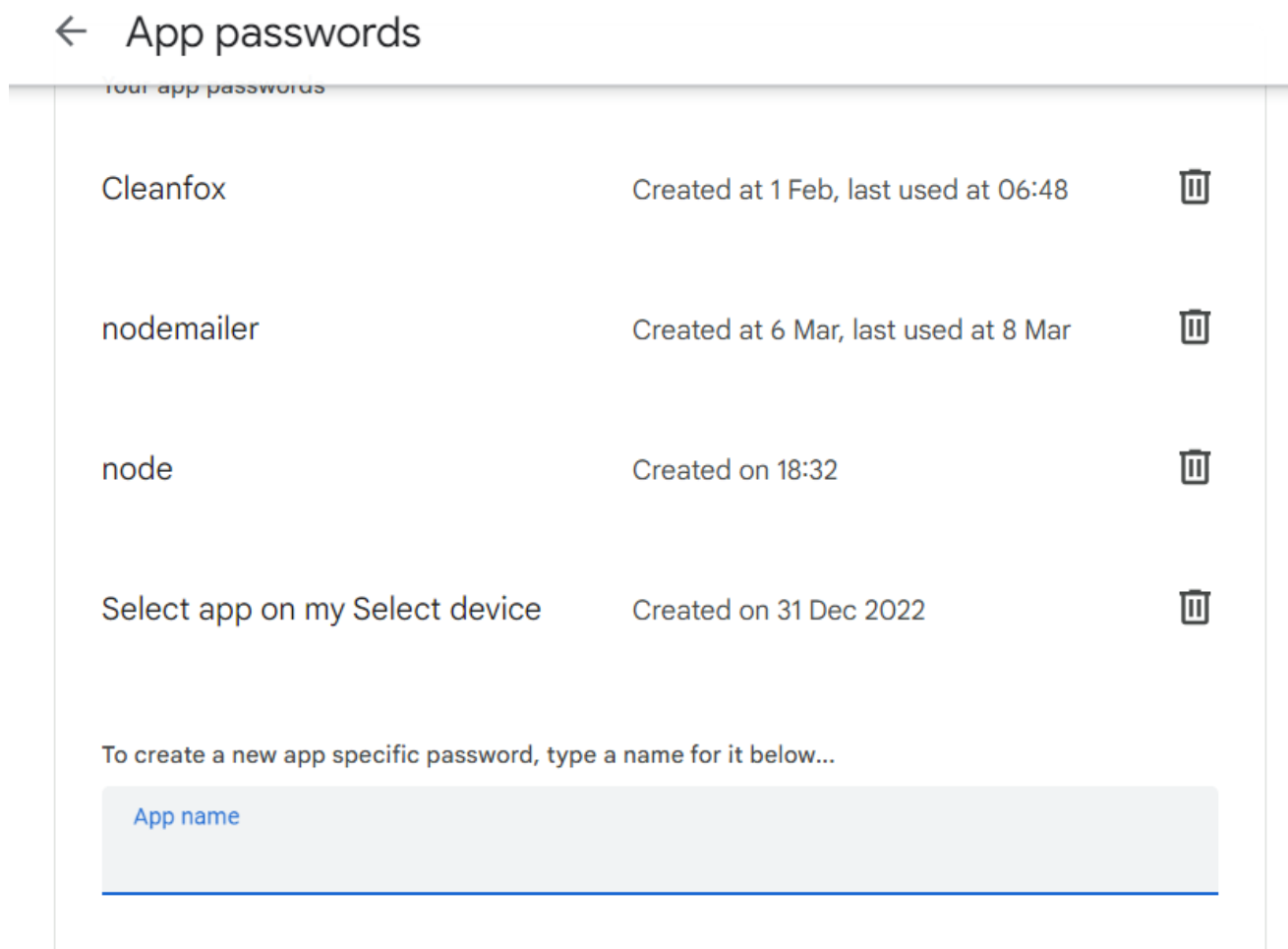
- Please configure the environment variable `school` with your school's name. This designated name will serve as the database name, ensuring a seamless integration with our database system.

Setting up Mail Server

1. **Establish a Dedicated Gmail Account for Credential Transmission:** To ensure the security and integrity of credential transmission, we recommend creating a Gmail account exclusively for this purpose.
2. **Access 'App Passwords' in Your Google Account Settings:** Navigate to 'Manage your Google Account' and search for 'App passwords'. Alternatively, you can access this feature directly through the following link: [Google Account - App Passwords](#) (Please ensure you are logged into the correct account).



3. **Generate a New Application Password:** Create a new application entry and securely copy the generated password.



4. **Configure Environment Variables for NodeMailer:** In your application's environment, set the `NODEMAIL` environment variable as the application name you just created. Additionally, set `NODEMAIL_PASSWORD` as the copied password. Be sure to remove any spaces from these values for accurate configuration.

By following these steps, you'll ensure a professional and secure approach to managing credentials for NodeMailer integration.

Setting up JWT Key

- Please establish the `JWT_KEY` environment variable by assigning it any string of your choice. This key will serve as a critical component in our security infrastructure.

Setting up MongoDB

Let's begin by setting up [MongoDB Atlas](#), a cloud-based database service that simplifies database management, especially during deployment. The MongoDB free tier offers ample space for starting out, with a generous shared cluster plan cap of 512MB.



Here's how you can set up MongoDB:

1. Logging in and Database access:

- After logging in, access the "Database Access" section under Security.
- Add a new database user with appropriate credentials.
- Choose "Atlas Admin" as the Built-in Role or select a role based on your requirements.

Add New Database User

Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding [Access Manager](#)

Authentication Method

Password

Certificate

AWS IAM
(MongoDB 4.4 and up)

PREVIEW
Federated Auth
(MongoDB 7.0 and up)

MongoDB uses [SCRAM](#) as its default authentication method.

Password Authentication

bloguser1

.....

SHOW

Autogenerate Secure Password

Copy

Database User Privileges

Configure role based access control by assigning database \$user a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. **You must choose at least one role or privilege.** [Learn more about roles.](#)

Built-in Role
Select one [built-in role](#) for this user.

Atlas admin

1 SELECTED

2. Network Access Configuration:

- Navigate to "Network Access" under Security.
- Add an IP address. During development, allowing all IPs (0.0.0.0/0) can be suitable. Remember to restrict IPs during deployment.

Add IP Access List Entry ✕

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)

ADD CURRENT IP ADDRESS

Access List Entry:

0.0.0.0/0

Comment:

Dev Stage



This entry is temporary and will be deleted in

6 hours

Cancel

Confirm

3. Copying driver code

- Go to the "Database" section under the "Deployment" tab.
- Click on "Connect" and select the "Driver" option.
- Choose "Node.js" as the driver and specify the version you're using.
- Copy the connection string provided and store it securely. This connection string will be used to establish a connection from your API to the database.

4. Setting up env variable

- Kindly replace 'username' and 'password' with your actual credentials. Following this, configure an environment variable named `MONGODB_URL` with the updated connection string for your MongoDB database. This will ensure seamless and secure access to the database

Setting up Firebase Storage

Firebase is a comprehensive mobile and web application development platform offered by Google. It provides a wide range of tools and services to help developers build high-quality apps quickly, without needing to manage the underlying infrastructure. Firebase is known for its ease of use, scalability, and integration with other Google Cloud services.



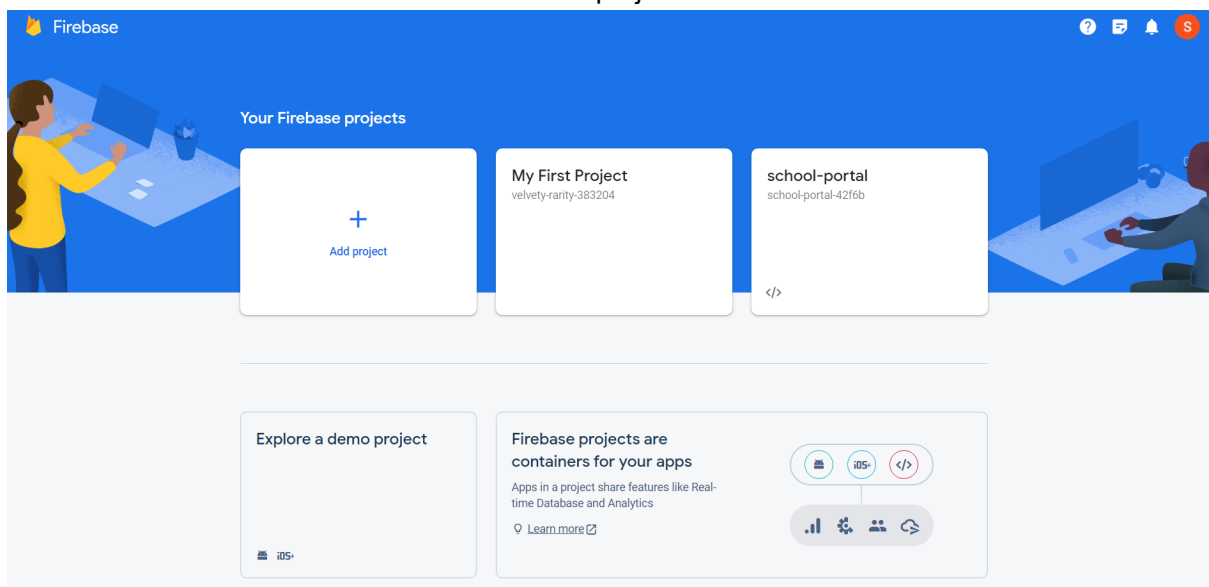
Firebase

1. Access Firebase:

- Visit the Firebase website at [Firebase](https://firebase.google.com/).

2. Create a New Project:

- Access the Firebase console and create a new project.

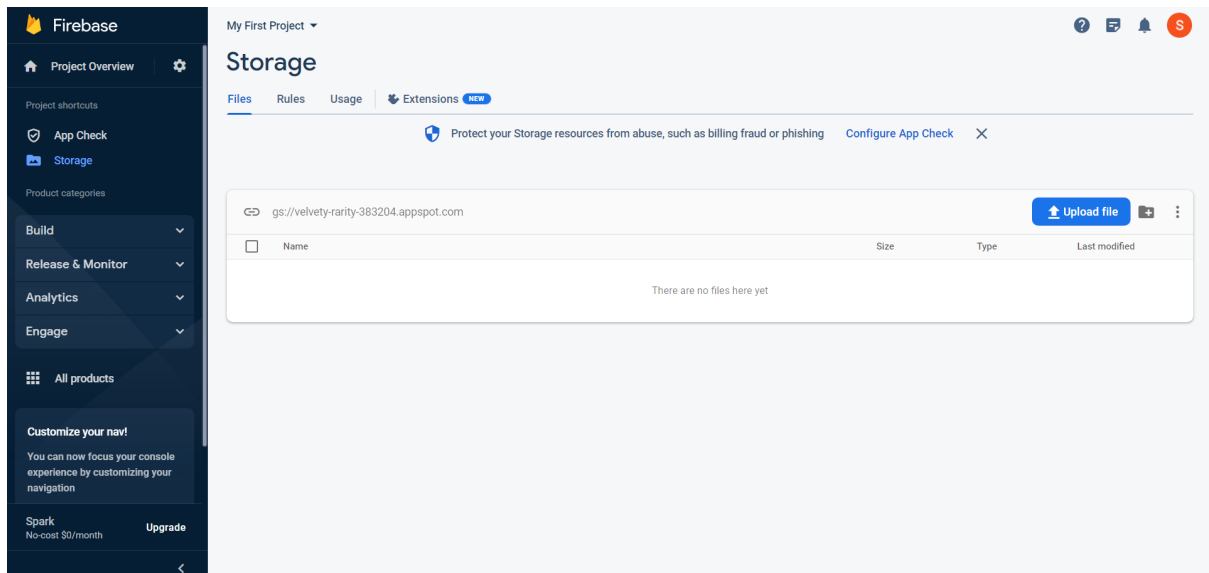


3. Select Firebase Storage:

- Within your project, select "Storage" from the list of available Firebase products.

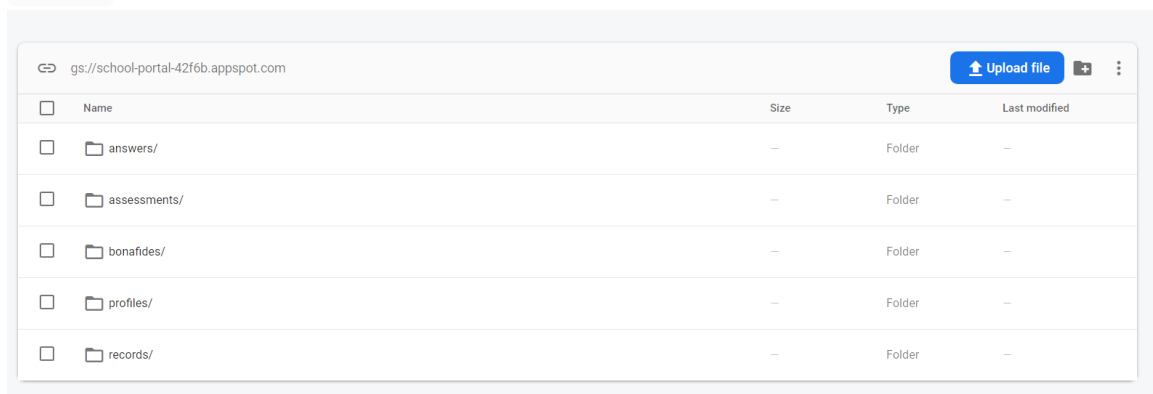
4. Get Started with Firebase Storage:

- Click "Get Started" to initiate the setup process. You'll be prompted to choose a mode and specify a location for your storage, which may take a moment to configure.



5. Create Folders:

- Once you're redirected to the storage bucket, create the following folders as needed:
 - answers
 - assessments
 - bonafides
 - profiles
 - records

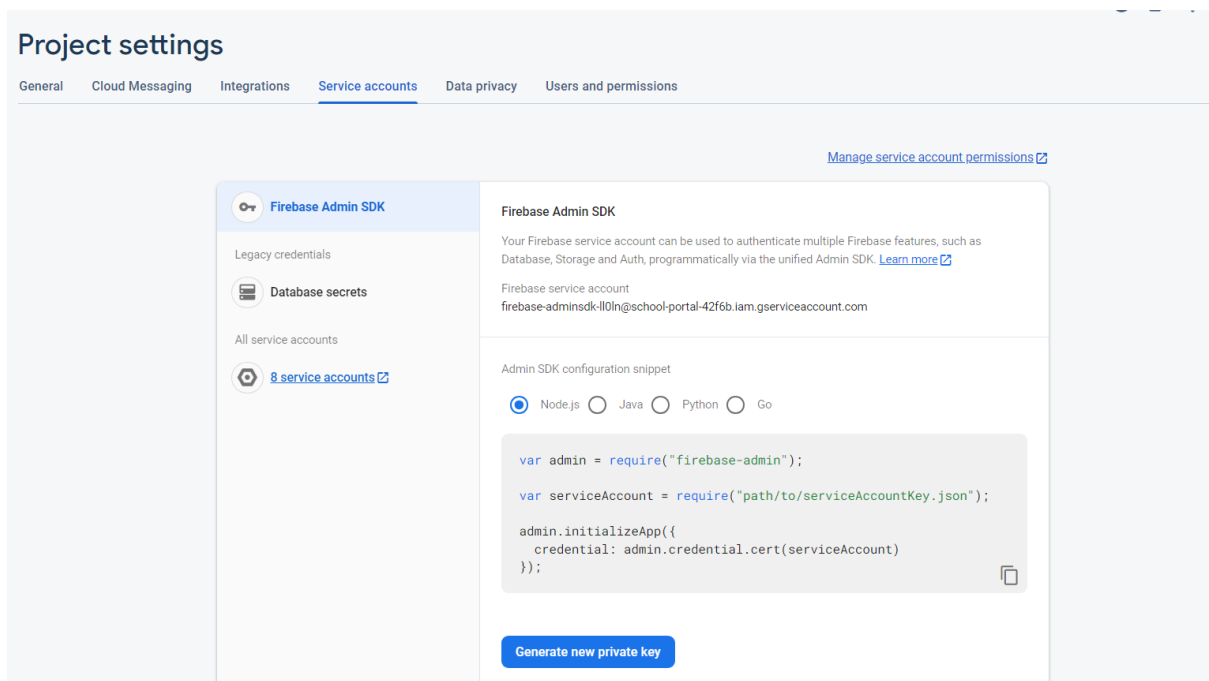


6. Generate Service Account Key:

- Navigate to your project settings and click on "Service Accounts."

7. Download a Private Key:

- Click on "Generate new private key." This action will download a JSON file containing the necessary credentials.



8. Configure Environment Variables:

- For added security and flexibility, it's advisable to set up environment variables using the values from the downloaded JSON file. Ensure that each key in the JSON file corresponds to a unique environment variable with the same value.

By following these steps, you'll establish a robust Firebase Storage system and maintain enhanced security by configuring essential credentials as environment variables. This approach simplifies management and provides a professional level of data security.

Setting up Razorpay

Razorpay is a developer-friendly payment portal which can easily integrate into code. Check out the Official Website: <https://razorpay.com/>



1. Sign Up and Prepare the Test Environment:

- Begin by signing up for a Razorpay account and navigate to 'Account & Settings.'
- Access 'API Keys' and generate new API Keys specifically for the test environment. Be sure to download them.
- When your server is ready for production, you can switch to 'Production Mode.' This switch requires the completion of a KYC form.

A screenshot of the Razorpay dashboard showing the 'API keys' section. The top navigation bar includes a search bar, a 'YOU'RE IN TEST MODE' toggle, and links for 'Exclusive Offer', 'Test Mode', and user profile. The main content area shows the 'API keys' tab selected, with a warning message: 'You are in Test Mode, so only test data is shown. [Activate your account](#) to start making live transactions.' Below this is a table with one API key entry.

Key Id	Created At	Expiry	Action
rzp_test_73wohYapKBYpbR	Aug 9th, 2023 08:14:30 PM	Never	Regenerate Test Key

2. Configure Environment Variables:

- To enable seamless integration, configure the following environment variables:
 - `rpy_key` : Set this variable to your Razorpay API key.
 - `rpy_key_secret` : Set this variable to your Razorpay API key secret.

3. Establish a Webhook for Backend Integration:

- You should set up a webhook for the backend.
- It's crucial to set up a webhook for your backend to stay informed about payment events.

- Configure the following details:
 - Webhook URL: This should be your production URL with '/transactions/razorpay' appended.
 - Secret: Add a secret for added security, and set it using the environment variable `razorpay_webhook_secret`.
 - Action Events: Specify 'payment.captured' as the action event to capture.

Webhook Setup

Webhook URL *

HTTPS URL is recommended

Secret

We strongly recommend adding secrets for security [🔗](#)[Show Secret](#)

Alert Email

sanjaynithin1910@gmail.com

Receive email alerts for webhook failures

Active Events *

Search

☐ Payment Events

☐ payment.authorized

☐ payment.failed

☒ payment.captured

☐ payment.dispute.created

☐ payment.dispute.won

☐ payment.dispute.lost

☐ payment.dispute.closed

1 event selected

[Clear all](#)

[Know more about the events](#) [🔗](#)

Cancel

Create Webhook

Setting up Production URL

- Please configure the environment variable `url` with the URL where the backend will be deployed. This URL serves as a critical reference point for ensuring seamless deployment and connectivity.