Creating a paperless app using Django for managing various forms and workflows within your company sounds like a great project. Django provides a robust framework for building web applications, and you can leverage its features to implement the desired functionality. Here's a step-by-step approach to help you get started:

**Define the requirements:** Begin by identifying all the forms and workflows that need to be digitized. Document the fields and steps involved in each process, along with the departments and personnel responsible for approval at each stage.

**Design the database schema:** Based on the requirements, design the database schema that will store the form data and related information. Identify the relationships between different models, such as Employee, Department, Form, and Workflow.

**Create Django models:** Define the necessary Django models based on the database schema you designed. For example, you might have models for Employee, Department, Form, Workflow, Approval, etc. Configure the relationships between these models using Django's model fields (e.g., ForeignKey, ManyToManyField).

**Implement user authentication and authorization:** Django provides built-in user authentication and authorization features. You can create user accounts for each employee, assigning them to their respective departments and units. Utilize Django's authentication system to handle user login, registration, and password management.

**Build forms:** Implement forms using Django's form functionality. Django forms allow you to define the fields, validation rules, and layout of the forms. You can create separate form classes for each type of form required within your application.

**Implement workflows:** Define the workflows for each form, specifying the steps involved and the approvers at each stage. You can represent workflows using Django models and implement the logic for moving forms through the workflow stages.

**Implement views and templates:** Create Django views to handle form submissions, display form details, and manage the workflow. Design templates to render the forms and workflow status for users. Django's template engine allows you to integrate dynamic data into your HTML templates.

**Configure routing and URLs:** Define the URL patterns in Django's URL configuration to map URLs to the appropriate views. This enables users to access different sections of your application.

**Handle notifications and email alerts:** Implement a mechanism to notify users when a form is assigned to them for approval or when there are updates in the workflow. Django provides email functionality to send notifications to users.

**Test and refine:** Thoroughly test your application to ensure that forms, workflows, and user permissions work as expected. Identify any issues or areas for improvement and refine your implementation accordingly.

**Deploy and host:** Deploy your Django application to a production server, ensuring it's accessible to your company's employees. Choose an appropriate hosting environment based on your company's requirements.

Remember, this is a high-level overview, and the actual implementation will depend on the specifics of your project. Throughout the development process, refer to the official Django documentation (<https://docs.djangoproject.com/>) for detailed information about the various Django features and best practices.

Good luck with your project!