perplexity

**create for me a full survey system using this framework like** kmsurveytool.com/app **this is a tool like kobotoolbox where the admin can create public surveys and polls and send the link to multiple recepients , anyone who has this link can fillout this survey, charts reporting and a dynamic form builder like kobo toolbox or google forms where it can have unlimited elements and unlimited questions, all questions can be reordered there are independent poll creators and organization poll creators too these two actors surveys can only be filled by registered respondents and for independent survey creators to join they must pay via paypal 1 dollar after which admin sets their account to active this is an annual subscription, every actor has their own database, it must have robust reporting with charts and can be exported to excel, one can view analytical reports both quantitative and qualitative, its an enterprise grade survey platoform that covers market analysis, political polls and academic surveys (esp for eg thesis students at the university) i will leave you to figure out the rest, make it world class, for the interface styling you can use tailwind and fontawesome icons some of the survey elements include, geolocation (respondent location at time of filling survey and for geo fencing) video, audio, Matrix questions (agree strongly agree 1-5 etc) text box text area, radio checkbox etc the survey form builder can be previewed before saving, one can fill partially and save a draft to come continue**

## filling later and once one is satisfied can save the form and set the survey as either public or private and be able to send the survey taking link to potential respondents via social media eg whatsapp and through mass email, proceed

Creating a full-fledged enterprise-grade survey system like kmsurveytool.com/app with advanced dynamic form building, multi-actor roles, subscription payments, detailed reporting, and modern UI using the provided lightweight Python microframework is a very large project. I'll outline a comprehensive system design and development plan first, followed by a modular approach and core implementation examples aligned with your requirements.

### Survey System Design Outline

### Actors/Roles

- **Admin**: Manages users, subscriptions, survey creators, survey approval, reports.
- **Independent Survey Creators**: Users who pay and are approved by admin, create own surveys.
- **Organization Survey Creators**: Users tied to organizations with their own survey pools.
- **Respondents**: Registered users who fill assigned surveys.

### Core Modules and Features

### 1. Authentication & Subscription

- User registration/login with role assignment.
- Subscription via PayPal for Independent survey creators (annual $1 charge).
- Admin dashboard to activate/deactivate accounts.

### 2. Dynamic Survey Form Builder

- Unlimited question elements: text, textarea, radio, checkbox, matrix, audio, video, geolocation, etc.
- Dynamic drag/drop reordering.
- Preview form before saving.
- Surveys saved as JSON schema for flexible form generation.
- Survey privacy: public or private (respondent-only, registration required).
- Save partially filled responses (drafts).

### 3. Survey Management

- Survey listing (public/private).
- Polls with limited questions.
- Survey URL generation for sharing (could be unique per survey).
- Multi-channel sharing: email integration, WhatsApp/social media links.
- Geo-fencing logic for surveys with location constraints.

### 4. Survey Respondent Experience

- Link-based access to surveys.
- Responsive form UI with validation.
- Draft saving and resuming responses.
- Submit final response.

### 5. Reporting and Analytics

- Dashboard with charts for survey results (quantitative and qualitative).
- Export reports to Excel/CSV.
- Cross-filtering and report drill-down.
- Heatmaps/geolocation analytics.

### 6. System Architecture

- Modular with RESTful routes.
- ORM models for users, surveys, questions, responses, payments.
- Middleware for auth, payment check, role enforcement.
- Async processing for better concurrency.
- Template-based (Tailwind CSS + FontAwesome) UI.
- Charting library integration (e.g., Chart.js).

## Development Plan (Phased)

### Phase 1: Foundation

- Base user system with roles and authentication.
- Survey model and basic CRUD.
- Simple dynamic form renderer from JSON schema.
- Survey taking interface.

### Phase 2: Subscription and Payments

- PayPal integration and subscription management.
- Admin management UI to approve creators.

### Phase 3: Survey Builder Enhancements

- Drag/drop question reordering.
- New question types (geo, multimedia).
- Draft saving.

### Phase 4: Reporting Module

- Aggregate data collection.
- Chart-based reports, export feature.
- Analytical dashboards.

### Phase 5: Sharing and Notifications

- Mass mailing module.
- Social media share links.

## Core Implementation Examples

### Model Definitions (ORM)

```
class User(Model):
    id = Field(int, unique=True, nullable=False)
    name = Field(str)
    email = Field(str, unique=True)
    password_hash = Field(str)
    role = Field(str)  # admin, independent_creator, org_creator, respondent
    subscription_active = Field(bool, default=False)
    subscription_expiry = Field(str)  # ISO date

class Survey(Model):
    id = Field(int, unique=True)
    creator_id = Field(int)
    title = Field(str)
    description = Field(str)
    json_schema = Field(str)  # JSON serialized questions
    is_public = Field(bool, default=True)
    status = Field(str)  # draft, active, closed
    created_at = Field(str)
    updated_at = Field(str)

class Response(Model):
    id = Field(int)
    survey_id = Field(int)
    respondent_id = Field(int)
    answers = Field(str)  # JSON serialized answers
    status = Field(str)  # in_progress, submitted
    created_at = Field(str)
    updated_at = Field(str)
```

### Survey Creation Route Example

```
@app.route("/survey/create", methods=["GET", "POST"])
@loginrequired
@rolesrequired(["independent_creator", "org_creator"])
async def create_survey(request):
    if request.method == "POST":
        data = await request.json
        # validate survey JSON schema
        # save survey in DB
        survey = Survey(
            creator_id=request.session.get("userid"),
            title=data.get("title"),
            description=data.get("description"),
            json_schema=json.dumps(data.get("questions")),
            is_public=data.get("is_public", True),
            status="draft",
            created_at=datetime.datetime.now().isoformat(),
            updated_at=datetime.datetime.now().isoformat()
        )
        survey.saveself(app)
        return Response({"message": "Survey saved", "survey_id": survey.id}, contenttype=
    else:
        content = app.templateengine.render("survey_create.html")
        return Response(content, contenttype="text/html")
```

### Survey Taking Interface (Dynamic Form Rendering)

- Load survey JSON schema.
- Generate HTML form dynamically using Tailwind CSS classes.
- Handle question types with custom widgets (text, checkbox, matrix, audio upload, geolocation).
- Save partial or final responses asynchronously.

### Reporting Module Snippet (Chart Data Retrieval)

```
@app.route("/survey/<int:id>/report")
@loginrequired
@rolesrequired(["admin", "independent_creator", "org_creator"])
async def survey_report(request, id):
    survey = Survey.find(id=id)[^0]
    responses = Response.find(survey_id=id, status="submitted")
    # Aggregate data, prepare stats
    data = process_responses_for_charts(responses)
    content = app.templateengine.render("survey_report.html", {"data": data})
    return Response(content, contenttype="text/html")
```

### Interface and Styling

- Use Tailwind CSS CDN in all templates.
- Use FontAwesome icons CDN.
- Responsive UX for form builder and survey taking.
- Drag and drop JS library for question reordering.
- Chart.js for charts in reporting.

This scope represents a complex, enterprise-ready survey system built on the light Python microframework you provided. Implementation will involve dozens of files, frontend and backend modules, and integration points.

If you want, I can help produce a partial or staged code example by module or implement a minimal viable product survey system step-by-step that you can extend and polish.

Would you prefer starting with a single module such as:

- User auth and subscription,
- Dynamic form builder,
- Survey taking interface,
- Reporting dashboard,
  or a simple MVP that demonstrates the core workflows first?

⁂

1. https://www.twilio.com/en-us/blog/design-phone-survey-system-python-google-sheets-twilio
2. https://www.youtube.com/watch?v=I0_nCZI4ug0
3. https://www.geeksforgeeks.org/python/online-survey-tool-using-django/
4. https://ploomber.io/blog/survey-python-frameworks/
5. https://www.stxnext.com/blog/beginners-introduction-python-frameworks
6. https://python.plainenglish.io/creating-a-smart-survey-system-using-chatgpt-python-and-postgresql-092efa3c5379
7. https://www.airweb.org/article/2018/09/20/tech-tip
8. https://www.dataquest.io/blog/how-to-analyze-survey-data-python-beginner/