**1. Define the Objective**

Create a system to detect influencers with fake followers or engagement, providing a "fraud likelihood" score.

**2. Data Collection**

1. **API Integration**:
   * Use APIs from Instagram, Twitter, or other platforms to collect influencer data (followers, likes, comments, posts, etc.).
   * Tools: Tweepy (for Twitter), Instaloader, or similar APIs.
2. **Web Scraping**:
   * Scrape influencer profiles for data if APIs are restricted. Use libraries like BeautifulSoup or Selenium.
3. **Example Data Points**:
   * Number of followers.
   * Engagement metrics (likes, comments, shares).
   * Growth rate (daily/weekly follower growth).
   * Follower demographics (if available).

**3. Data Preprocessing**

1. **Clean and Normalize Data**:
   * Handle missing values.
   * Normalize numeric metrics (e.g., likes-to-followers ratio).
2. **Feature Engineering**:
   * Create metrics like:
     + **Engagement rate**: (Total Likes + Total Comments) / Followers.
     + **Follower growth rate**.
     + **Ratio of suspicious comments or bot-like text patterns**.
3. **Labeling**:
   * Use existing datasets or heuristics to label data (e.g., influencers flagged as fraudulent vs. genuine).

**4. Exploratory Data Analysis (EDA)**

1. Analyze patterns in the data:
   * Distribution of engagement rates.
   * Correlation between followers and engagement.
2. Visualize anomalies, such as unusually high follower growth in a short period.

**5. Model Development**

1. **Choose Models**:
   * Classification models (e.g., Logistic Regression, Random Forest, Gradient Boosting).
   * Anomaly detection models (e.g., Isolation Forest, DBSCAN) for unsupervised detection.
2. **Train-Test Split**:
   * Split your dataset into training, validation, and testing sets.
3. **Evaluation Metrics**:
   * Use precision, recall, and F1 score to evaluate model performance.

**6. Build the Web Application**

1. **Backend**:
   * Use Flask or FastAPI to serve your model.
   * Implement endpoints for data input (e.g., influencer handles) and result output (fraud likelihood score).
2. **Frontend**:
   * Create a user-friendly interface with frameworks like Streamlit or a simple HTML/CSS/JavaScript setup.
   * Display metrics, scores, and visualizations (e.g., fraud likelihood, engagement trends).
3. **Visualization**:
   * Use libraries like Plotly or D3.js for interactive charts.

**7. Deployment on Heroku**

1. **Prepare for Deployment**:
   * Package your application (e.g., requirements.txt, Procfile).
   * Use a lightweight database like SQLite for deployment or connect to cloud databases (e.g., PostgreSQL on Heroku).
2. **Deploy**:
   * Push your app to Heroku using Git.
   * Ensure scalability by enabling Heroku dynos if necessary.

**8. Post-Deployment Testing**

1. Test with real influencer handles.
2. Monitor app performance and gather user feedback.
3. Iterate based on feedback to improve detection accuracy.