Complete Setup Guide - Feature Voting System (Windows)

Part 1: Flask Backend Setup

Prerequisites

1. **Python 3.8+**
   - Download from python.org
   - Make sure to check "Add Python to PATH" during installation
2. **PostgreSQL**
   - Download from postgresql.org
   - During installation, remember your postgres user password

Step 1: Setup PostgreSQL Database

1. Open **pgAdmin** (installed with PostgreSQL)
2. Connect to your PostgreSQL server
3. Create a new database called `feature_voting`
4. Open Query Tool and run the SQL schema from the artifact

Step 2: Setup Flask Backend

```bash
# Create project directory
mkdir feature-voting-system
cd feature-voting-system

# Create backend directory
mkdir backend
cd backend

# Create virtual environment
python -m venv venv

# Activate virtual environment
venv\Scripts\activate

# Install dependencies (save the requirements.txt file first)
pip install -r requirements.txt
```

Step 3: Environment Configuration

Create a `.env` file in the backend directory:

```env
DATABASE_URL=postgresql://postgres:your_password@localhost/feature_voting
SECRET_KEY=your-super-secret-key-change-this
FLASK_ENV=development
```

Step 4: Run Flask Backend

```bash
# Make sure you're in backend directory and virtual environment is active
python app.py
```

The API will be available at `http://localhost:5000`

Step 5: Test the API

Use these curl commands or Postman:

```bash
# Test health endpoint
curl http://localhost:5000/api/health

# Get features
curl http://localhost:5000/api/features

# Create a user (needed for voting)
curl -X POST http://localhost:5000/api/users \
  -H "Content-Type: application/json" \
  -d '{"username": "testuser", "email": "test@example.com", "password": "password123"}'

# Vote for a feature (replace feature_id and user_id)
curl -X POST http://localhost:5000/api/features/1/vote \
  -H "Content-Type: application/json" \
  -d '{"user_id": 1}'
```

Part 2: Android Studio Setup (For Future Development)

Prerequisites for Android Development

1. **Java Development Kit (JDK)**
   - Download JDK 17 from Oracle or OpenJDK
2. **Android Studio**
   - Download from developer.android.com
   - This is a large download (~1GB+)

Android Studio Installation Steps

1. Run Android Studio installer
2. Follow setup wizard
3. Install Android SDK (API level 33 or latest)
4. Create or configure Android Virtual Device (AVD)

Create Android Project

1. Open Android Studio
2. Create New Project
3. Choose "Empty Compose Activity"
4. Set:
   - Name: Feature Voting
   - Package: com.example.featurevoting
   - Language: Kotlin
   - Minimum SDK: API 24
5. Wait for project to sync

Part 3: EASIER TESTING APPROACH - Web Frontend

Since Android development has a learning curve, I recommend starting with a simple web frontend to test your Flask API:

Simple HTML Test Interface

Create `test_frontend.html` in your backend directory:

html

```
<!DOCTYPE html>
<html>
<head>
    <title>Feature Voting Test</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .feature { border: 1px solid #ddd; padding: 10px; margin: 10px 0; }
        button { padding: 5px 10px; margin: 5px; }
    </style>
</head>
<body>
    <h1>Feature Voting System Test</h1>
    <button onclick="loadFeatures()">Load Features</button>
    <div id="features"></div>

    <script>
        const API_BASE = 'http://localhost:5000/api';
        const USER_ID = 1; // Test user ID

        async function loadFeatures() {
            try {
                const response = await fetch(`${API_BASE}/features`);
                const data = await response.json();
                displayFeatures(data.features);
            } catch (error) {
                console.error('Error:', error);
                document.getElementById('features').innerHTML = 'Error loading features';
            }
        }

        async function voteFeature(featureId) {
            try {
                const response = await fetch(`${API_BASE}/features/${featureId}/vote`, {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ user_id: USER_ID })
                });
                if (response.ok) {
                    loadFeatures(); // Reload to show updated vote count
                } else {
                    const error = await response.json();
                    alert(error.error);
                }
            } catch (error) {
                console.error('Error voting:', error);
            }
        }

        function displayFeatures(features) {
            const container = document.getElementById('features');
            container.innerHTML = '';

            features.forEach(feature => {
                const div = document.createElement('div');
                div.className = 'feature';
                div.innerHTML = `
                    <h3>${feature.title}</h3>
                    <p>${feature.description}</p>
                    <p>Author: ${feature.author} | Votes: ${feature.vote_count}</p>
                    <button onclick="voteFeature(${feature.id})">Vote</button>
                `;
                container.appendChild(div);
            });
        }
    </script>
</body>
</html>
```

Test the Complete System

1. Start your Flask backend: `python app.py`
2. Open `test_frontend.html` in your web browser
3. Click "Load Features" to see the feature list
4. Click "Vote" buttons to test voting functionality

Troubleshooting Common Issues

Flask Issues:

- **Database connection error**: Check PostgreSQL is running and credentials in .env are correct
- **Port already in use**: Change port in app.py: `app.run(port=5001)`
- **CORS errors**: Make sure Flask-CORS is installed and configured

PostgreSQL Issues:

- **Connection refused**: Ensure PostgreSQL service is running
- **Authentication failed**: Double-check username/password in DATABASE_URL

Python Issues:

- **Module not found**: Make sure virtual environment is activated
- **Permission errors**: Run command prompt as administrator if needed

---

Next Steps After Backend is Working

1. **Test all API endpoints** using the HTML interface or Postman
2. **Add more features** to the database through the API
3. **Once comfortable**, proceed with Android Studio setup
4. **Start with simple Android tutorials** before implementing the full Jetpack Compose UI

Quick Verification Checklist

- [ ] PostgreSQL installed and running
- [ ] Database `feature_voting` created with schema
- [ ] Flask backend starts without errors
- [ ] API endpoints respond correctly
- [ ] Sample data visible in database
- [ ] Web interface can load and vote on features

This approach lets you verify the complete backend functionality before diving into Android development!