Recycling Platform Backend 🌱 ♻️

Capstone Project - Easy Development Setup

Quick Start for Team Members 🚀

1. Clone and Install

```bash
git clone https://github.com/your-username/recycling-platform-backend.git
cd recycling-platform-backend
npm install
```

2. Get Firebase Service Account Key

Ask team lead for the `serviceAccountKey.json` file and place it in the `config/` folder.

That's it! Everything else is already configured.

3. Start Development

```bash
npm run dev
```

Visit: http://localhost:3000/health (should show "OK")

Project Structure 📂

```
recycling-platform-backend/
├── models/           # All Firestore models (User, Post, etc.)
├── services/         # Auth, Storage, Notifications
├── config/           # Firebase configuration
├── uploads/          # Local file storage
├── .env              # ✅ COMMITTED (development config)
└── server.js         # Express server
```

API Endpoints 🛠️

Authentication

- `POST /api/auth/register` - Create new user
- `GET /api/protected/profile` - Get user profile (requires auth)
- `PUT /api/protected/profile` - Update profile (requires auth)

Posts

- `GET /api/protected/posts` - Get posts (with filters)
- `POST /api/protected/posts/waste` - Create waste post
- `POST /api/protected/posts/forum` - Create forum post
- `POST /api/protected/posts/initiative` - Create initiative post

File Uploads

- `POST /api/protected/upload/application-documents` - Upload application docs
- `POST /api/protected/upload/proof-of-pickup` - Upload pickup proof
- `GET /uploads/*` - Access uploaded files

Admin (requires Admin role)

- `GET /api/admin/users` - Get all users
- `GET /api/admin/storage-stats` - Get storage statistics

Model Usage Examples 💻

Create a Waste Post

javascript

```
const WastePost = require('./models/WastePost');

const wastePost = await WastePost.create({
  userID: 'user123',
  title: 'Plastic bottles ready for pickup',
  description: '50 clean PET bottles available',
  location: 'Manila, Philippines',
  items: [
    {
      itemName: 'PET Bottles',
      materialID: 'pet_bottles',
      sellingPrice: 15.50,
      kg: 2.5
    }
  ]
});

// Use built-in methods
console.log('Total value:', wastePost.getTotalValue());
console.log('Total weight:', wastePost.getTotalWeight());
```

Handle Pickup Workflow

javascript

```
const Pickup = require('./models/Pickup');

// Create pickup request
const pickup = await Pickup.create({
  postID: 'post123',
  giverID: 'giver456',
  collectorID: 'collector789',
  pickupTime: new Date('2024-02-01T10:00:00Z'),
  pickupLocation: 'Quezon City'
});

// Workflow progression
await pickup.confirm();      // Giver confirms
await pickup.complete({      // Collector completes
  itemName: 'PET Bottles',
  materialIDs: ['pet_bottles'],
  price: 38.75,
  kg: 2.5
}, 'proof-image-url');

// Both users automatically get points!
```

File Storage 📁

Currently using **local file storage** (no external service needed):

- Files saved to `uploads/` folder
- Accessible via `http://localhost:3000/uploads/filename`
- Automatic cleanup of temp files
- Easy to migrate to cloud storage later

Development Features 🔧

Built-in Validation

```
javascript
```

```
// All models have validation
try {
  const user = await User.create({
    firstName: '', // ❌ Will throw validation error
    email: 'invalid-email' // ❌ Will throw validation error
  });
} catch (error) {
  console.log(error.message); // Shows specific validation errors
}
```

Smart Inheritance

```javascript
// Post inheritance works automatically
const post = await Post.findById('post123');

// Returns correct subclass based on postType
if (post.postType === 'Waste') {
  post.getTotalValue(); // ✅ WastePost method available
}
```

Business Logic Methods

```javascript
// Rich methods for common operations
await user.addPoints(50, 'Post_Creation');
await user.addBadge('first_post_badge');
await pickup.confirm();
await material.updatePrice(25.00);
```

Testing 🧪

Manual API Testing

```bash
```

```
# Test user creation
curl -X POST http://localhost:3000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "firstName": "Test",
    "lastName": "User",
    "email": "test@example.com",
    "userType": "Giver"
  }'
```

Model Testing in Node REPL

```bash
node
> const User = require('./models/User');
> const testUser = await User.create({firstName: 'Test', lastName: 'User', email: 'test@test.com', userType: 'Giver'});
> console.log(testUser.userID);
```

Team Workflow 👥

Daily Development

```bash
git pull origin main    # Get latest changes
npm run dev            # Start development server
# Make changes...
git add .
git commit -m "feat: add new feature"
git push origin main   # Everyone shares same branch for capstone
```

Adding New Features

1. **Models first** - Add new fields or methods to model files
2. **Test models** - Use Node REPL to test model methods
3. **Add API routes** - Create Express routes in server.js
4. **Test API** - Use Postman or curl to test endpoints

Capstone Project Notes 📚

Why This Setup is Perfect for Capstone:

- ✅ **Zero external service dependencies**
- ✅ **Easy teammate onboarding** (just clone and run)
- ✅ **No billing surprises**
- ✅ **Professional code structure**
- ✅ **Easy to demo** to professors/judges
- ✅ **Production-ready foundation** (can easily upgrade later)

What Makes This Professional:

- 🏗 **Clean architecture** with separated models, services, routes
- 🛡 **Built-in validation** prevents bad data
- 📊 **Rich business logic** (points, badges, notifications)
- 🔄 **Proper inheritance** implementation
- 📖 **Self-documenting** code with clear model structure

When to Upgrade (Post-Capstone):

- Move to cloud file storage (Cloudinary/AWS)
- Add proper environment separation (dev/staging/prod)
- Implement proper security practices
- Add comprehensive testing suite

This gives you a professional, working backend that's perfect for capstone development and impressive for presentations! 🎓 ✨