Team Setup Guide 👥

For New Team Members

1. Clone the Repository

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

2. Install Dependencies

```bash
npm install
```

3. Get Required Secret Files

Option A: From Team Lead (Recommended)

Ask the team lead to securely share these files:

- `serviceAccountKey.json` → Place in `config/` folder
- `.env` file with all values filled → Place in root directory

Option B: Create Your Own Firebase Project (For Testing)

1. Go to Firebase Console
2. Create new project: `recycling-platform-dev-[yourname]`
3. Enable Firestore, Authentication
4. Download `serviceAccountKey.json` → Place in `config/`
5. Copy `.env.example` to `.env` and fill in your Firebase config

Option C: Get Added to Main Firebase Project

Ask team lead to add your Google account to the main Firebase project:

1. Go to Firebase Console → Project Settings → Users and permissions
2. Add your email with "Editor" role
3. Download your own `serviceAccountKey.json`

4. Create Required Directories

```bash
mkdir -p uploads/temp
mkdir -p uploads/applications
mkdir -p uploads/pickups
mkdir -p uploads/badges
mkdir -p uploads/profiles
```

5. Start Development Server

```bash
npm run dev
```

6. Test the Setup

Visit: `http://localhost:3000/health` Should return: `{"status":"OK","timestamp":"...","environment":"development"}`

For Team Lead / Project Setup

1. Firebase Project Setup

1. Create Firebase project
2. Enable Firestore Database (production mode)
3. Enable Authentication (Email/Password)
4. Download configuration and service account key

2. Sharing Secret Files Securely

Option A: Password Manager (Recommended)

Use team password manager (Bitwarden, 1Password, etc.):

1. Upload `serviceAccountKey.json` as secure note
2. Share `.env` contents as secure note
3. Share vault access with team members

Option B: Encrypted Archive

```bash
```

```
# Create encrypted zip
zip -e secrets.zip serviceAccountKey.json .env
# Share password via separate secure channel
```

Option C: Environment-Specific Setup

Create separate Firebase projects:

- `recycling-platform-dev` (development)
- `recycling-platform-staging` (testing)
- `recycling-platform-prod` (production)

Give team members access to dev project only.

3. Repository Setup

```bash
# Initialize git (if not done)
git init
git add .
git commit -m "Initial project setup with Firestore models"
git branch -M main
git remote add origin https://github.com/your-username/recycling-platform-backend.git
git push -u origin main
```

4. Team Member Onboarding Checklist

- [ ] Add to Firebase project (if using shared project)
- [ ] Share secret files securely
- [ ] Add to GitHub repository
- [ ] Share team coding standards/conventions
- [ ] Provide API documentation
- [ ] Set up development environment

Security Best Practices 🔐

Why These Files Are Secret

- **`serviceAccountKey.json`**: Contains private keys that give **full admin access** to your Firebase project
- **`.env`**: Contains API keys and secrets that could be exploited

What Could Go Wrong

If these files are committed to GitHub:

- ❌ Anyone can download your database
- ❌ Delete all your data
- ❌ Impersonate users
- ❌ Send spam notifications
- ❌ Rack up Firebase charges on your account

Safe Sharing Methods

✅ **Password managers** (Bitwarden, 1Password) ✅ **Encrypted files** with password shared separately
✅ **Secure team chat** (encrypted channels) ✅ **In-person/video call** sharing

❌ **Never share via:**

- Email attachments
- Slack/Discord direct messages
- Public GitHub repositories
- Shared Google Drive folders
- Text messages

File Storage Strategy 📁

Current Setup: Local Storage

Pros:

- ✅ No additional costs
- ✅ Simple setup
- ✅ Works for development and small teams
- ✅ No external dependencies

Cons:

- ❌ Files lost if server restarts (on platforms like Heroku)
- ❌ No CDN for fast global access
- ❌ Manual backup required

- ❌ Limited scalability

Future Migration Options

When Ready to Scale:

1. **Cloudinary** (Recommended)
   - 25GB free tier
   - Image optimization
   - Easy migration from local storage
2. **AWS S3**
   - 5GB free for 12 months
   - Highly scalable
   - Industry standard
3. **Supabase Storage**
   - 1GB free
   - Simple API
   - Good for small projects

Migration Strategy:

```javascript
// When ready, just change the storageService.js import
// From: require('./services/storageService') // Local
// To:   require('./services/cloudinaryService') // Cloudinary
// All your model code stays the same!
```

Development Workflow 🔄

1. Daily Development

```bash
```

```
git pull origin main
npm run dev
# Make changes
git add .
git commit -m "feat: add new feature"
git push origin feature-branch
```

2. File Upload Testing

Use tools like Postman or curl:

```bash
curl -X POST \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -F "documents=@test-file.pdf" \
  -F "applicationID=test-app-id" \
  http://localhost:3000/api/protected/upload/application-documents
```

3. Database Testing

```javascript
// Test in Node.js REPL
const User = require('./models/User');

// Create test user
const testUser = await User.create({
  firstName: 'Test',
  lastName: 'User',
  email: 'test@example.com',
  userType: 'Giver'
});

console.log('Created user:', testUser.userID);
```

Troubleshooting 🔧

Common Issues

"Firebase Admin not initialized"

- Check if `serviceAccountKey.json` exists in `config/` folder
- Verify file permissions (readable by Node.js)
- Check Firebase project ID in `.env`

"Permission denied" on file uploads

```bash
# Fix upload directory permissions
chmod 755 uploads/
chmod 755 uploads/temp/
```

"CORS error" from frontend

- Add your frontend URL to `ALLOWED_ORIGINS` in `.env`
- Check if frontend is sending credentials

Port already in use

```bash
# Kill process using port 3000
lsof -ti:3000 | xargs kill -9
# Or use different port
PORT=3001 npm run dev
```

Getting Help

1. Check this guide first
2. Look at error logs in terminal
3. Ask in team chat with error details
4. Create GitHub issue if it's a bug

Next Steps After Team Approval ✨

1. Enhanced Features

- Add email notifications
- Implement real-time updates with Firestore listeners
- Add image compression and resizing

- Implement caching layer

2. Production Setup

- Set up production Firebase project
- Configure proper security rules
- Add monitoring and logging
- Set up CI/CD pipeline

3. External Services Integration

- Choose file storage provider (Cloudinary recommended)
- Add email service (SendGrid, Nodemailer)
- Add SMS notifications (Twilio)
- Add analytics (Google Analytics, Mixpanel)

This setup gives you a solid foundation that can grow with your team! 🌱