VarsityVibe — Full Laravel Plan (no SSO)

Project name: varsityvibe

Goal: Build a VarsityVibe-style student discount platform entirely in Laravel (frontend + backend) and allow Nextschool users to log in via API credential verification (no SSO).

1. Executive summary

- Full Laravel monolith serving Blade frontend and backend admin/merchant UI.
- Nextschool integration is API-based credential verification: VarsityVibe will call Nextschool to validate login credentials and then create/update a local user record.
- Features: student verification, merchant onboarding, product/offers, admin controls, public catalogue, API endpoints for Nextschool to consume products if needed.

2. High-level flow diagram

```
[Visitor] --> VarsityVibe (Login Form)
|
v
VarsityVibe -> Nextschool API (POST /api/check-user)
|
(Nextschool validates credentials and returns user data)
|
v
VarsityVibe creates/updates local `users` record
|
v
Laravel Auth session started
|
User can browse offers (if verified)
```

3. Key routes (routes/web.php)

```
// Public
Route::get('/', [HomeController::class, 'index']);
Route::get('/merchants', [MerchantPublicController::class, 'index']);
Route::get('/merchants/{slug}', [MerchantPublicController::class, 'show']);
Route::get('/offers/{id}', [OfferController::class, 'show']);

// Auth
Route::get('/login', [AuthController::class, 'showLoginForm'])->name('login');
Route::post('/login', [AuthController::class, 'login']);
Route::post('/logout', [AuthController::class, 'logout'])->name('logout');

// Merchant (protected)
Route::middleware('auth','can:merchant')->group(function(){
Route::get('/merchant/dashboard', [MerchantController::class, 'dashboard']);
Route::resource('/merchant/products', MerchantProductController::class);
});

// Admin
Route::middleware(['auth','can:admin'])->prefix('admin')->group(function(){
Route::get('/', [AdminController::class, 'dashboard']);
Route::resource('/merchants', AdminMerchantController::class);
Route::resource('/student-verifications', StudentVerificationController::class);
});
```

4. API routes (routes/api.php) — for Nextschool integration

```
Route::post('/nextschool/verify-login', [Api.NextschoolController.class, 'verifyLogin']);
Route::post('/nextschool/sync-user', [Api.NextschoolController.class, 'syncUser']);
Route::get('/products', [Api.ProductController.class, 'index']);
```

Security: protect API endpoints with an API key (in header) and rate limit.

5. Example Controllers (snippets)

AuthController (important method)

```
use Illuminate Support Facades Http;
use Illuminate Support Facades Auth;

public function login(Request $request)
{
$request->validate([
'email' => 'required|email',
'password' => 'required',
]);

// Call Nextschool API to validate credentials
$resp = Http::withHeaders([
'X-API-KEY' => config('services.nextschool.api_key')
])->post(config('services.nextschool.url') . '/api/check-user', [
'email' => $request->email,
'password' => $request->password,
]);

if ($resp->failed()) {
return back()->withErrors(['email' => 'Invalid credentials.']);
}

$data = $resp->json();

$user = User::updateOrCreate(
['nextschool_user_id' => $data['id']],
[
'name' => $data['name'] ?? $data['email'],
'email' => $data['email'],
// If you prefer not to store password, create a random password locally
'password' => Hash::make(Str::random(32)),
'role' => $data['role'] ?? 'student',
]
);

Auth::login($user);
return redirect()->intended('/');
}
```

Notes: don't store the plain-text password; rely on Nextschool for validation. You may optionally store a hashed placeholder password.

6. Models & migrations (overview)

- User model: add nextschool_user_id (nullable), verified_student_at (nullable), role.
- Merchant model + migration
- Product model + migration
- Offer model + migration
- StudentVerification model + migration

Example migration for users:

```
Schema::table('users', function (Blueprint $table) {
$table->string('role')->default('student');
$table->unsignedBigInteger('nextschool_user_id')->nullable()->index();
$table->timestamp('verified_student_at')->nullable();
});
```

7. Student verification flow (admin approve)

1. Student signs up and uploads proof (student card or university email verification).
2. Record saved in student_verifications with status = pending.
3. Admin views pending verifications, checks documents, clicks Approve.
4. System sets student_verifications.status = approved and users.verified_student_at = now().

8. Merchant onboarding and product lifecycle

- Merchant registration form (admin approval recommended).
- Merchant creates product and offer.
- Admin can flag/disable offers.
- Public site shows only active offers and products.

9. Environment variables (add to .env)

```
NEXTSCHOOL_API_URL=https://nextschool.example.com
NEXTSCHOOL_API_KEY=long-secret-key
APP_NAME=VarsityVibe
```

10. Deployment & Dev notes

- Use Redis for cache and session in production.
- Use Supervisor to run php artisan queue:work for background jobs (emails, analytics).
- Use Cloud storage (S3) for merchant logos and student verification docs.

11. Deliverables included here
1. Flow + architecture (this document).
2. Laravel skeleton code: routes, controller snippets, migration examples (above).
3. Ready-to-run checklist and .env keys.

12. Next actions I will do now
- Generate a downloadable PDF of this plan and place it at /mnt/data/varsityvibe_plan.pdf for you to download.
- Scaffold a minimal Laravel project skeleton (file list and sample files) and add it to this canvas.

If you want additional customizations in the skeleton (for example: specific user fields, coupon system, or WhatsApp-based merchant onboarding), tell me now and I’ll include them in the scaffold.