**Universal Code Transpiler with UIR (Universal Intermediate Representation)**

**Project Requirements Document**

**1. Overview**

We are building a **universal code transpiler** that takes in **full project folders** (including frontend, backend, or both) and converts them into other frameworks/languages while preserving **UI, logic, and state**. The core is a **Universal Intermediate Representation (UIR)** that abstracts project structures into a standard form that can then be recompiled into any supported stack.

**2. Key Features**

1. **Full Project Upload**
   * Accept zipped project folders or GitHub repository links.
   * Detect the tech stack automatically (React, Kotlin, Flutter, Node.js, Python, Express, etc.).
   * Show **language/framework distribution** visually (pie chart like GitHub).
2. **Universal Intermediate Representation (UIR)**
   * Convert uploaded projects into a UIR format.
   * UIR must capture:
     + UI layout (component tree, styles)
     + Business logic (functions, classes, state handling)
     + API calls and endpoints
     + File structure and module relationships
   * UIR should be **language/framework agnostic**.
3. **Code Transpilation**
   * Convert UIR into any **supported language/framework**.
   * Maintain functionality, state, and logic.
   * Preserve responsive design and styling.
   * Example transformations:
     + React → Kotlin
     + Kotlin → React
     + React → Flutter
     + Flutter → React
     + Kotlin → Flutter
     + Flutter → Kotlin
     + Node.js → Python
     + Python → Node.js
     + Express → FastAPI
     + and more…
4. **Output**
   * Provide downloadable converted project folders.
   * Output must be runnable without manual patching.
5. **Category Detection**
   * Classify detected languages/frameworks into categories:
     + **Frontend/UI Development**: React, React Native, Flutter, Kotlin (Compose), SwiftUI, Angular, Vue, etc.
     + **Backend/API Development**: Node.js, Express, Python (FastAPI/Django/Flask), Go, Rust, PHP, etc.
     + **Full-stack Frameworks**: Next.js, Remix, Nuxt.js, Laravel, Spring Boot, etc.
     + **System/Low-level**: C, C++, Rust, Zig.
     + **Scripting/Data**: Python, R, Julia, Bash, Perl.
   * Store classification in UIR metadata.
6. **Conversion Matrix**
   * Maintain a JSON mapping of all **possible conversion pairs**.
   * Each conversion pair counts separately (e.g., React→Flutter is one, Flutter→React is another).
7. **Architecture**
   * **Frontend**: React + TailwindCSS for UI (upload, progress, results view).
   * **Backend**: Python (FastAPI) or Node.js (Express) for API.
   * **Core Transpiler Engine**: Implement AST parsing for each language.
   * **UIR Generator**: Converts AST into standard UIR JSON format.
   * **UIR Compiler**: Converts UIR back into target language AST, then source code.
8. **UI/UX**
   * Clean, modern design with animations.
   * GitHub-like repo analysis visualization.
   * Step-by-step progress indication.
9. **Non-Goals for Phase 1**
   * No LLM-based code translation in first iteration (pure AST + compiler approach).
   * No partial-project migration — only full folders initially.

**3. Workflow**

1. **Upload Project**
2. **Detect Languages/Frameworks**
3. **Generate UIR**
4. **Select Target Language/Framework**
5. **Compile UIR into Target**
6. **Download Converted Project**

**4. Example UIR Structure**

{

"metadata": {

"name": "Sample Project",

"original\_stack": ["React", "Node.js"],

"detected\_category": ["Frontend", "Backend"],

"file\_count": 48

},

"components": [

{

"name": "NavBar",

"type": "UIComponent",

"props": ["title", "links"],

"children": [

{ "type": "Text", "value": "Home" },

{ "type": "Link", "href": "/about" }

],

"styles": {

"backgroundColor": "#fff",

"padding": "10px"

}

}

],

"logic": [

{

"name": "fetchUserData",

"type": "Function",

"params": ["userId"],

"returns": "User",

"body": [

"const res = await fetch(`/api/users/${userId}`);",

"return await res.json();"

]

}

],

"apis": [

{

"method": "GET",

"endpoint": "/api/users/:id",

"handler": "getUserById"

}

]

}

**5. Tech Stack**

* **Frontend**: React + TailwindCSS
* **Backend**: FastAPI (Python)
* **Core Transpiler**: Python for AST parsing
* **Storage**: Supabase (project storage + metadata)
* **Auth**: Supabase Auth

**6. Future Enhancements**

* Integrate LLM-based refinement to handle framework-specific quirks.
* Support partial module migrations.
* Add CLI tool for offline use.
* Version diff visualization (before/after).