

**Edu-Path - by E-falcons**

The **EduPath** platform aims to help young Australians make informed education and training decisions while providing insights to the government. Let’s break down what your prototype should include:

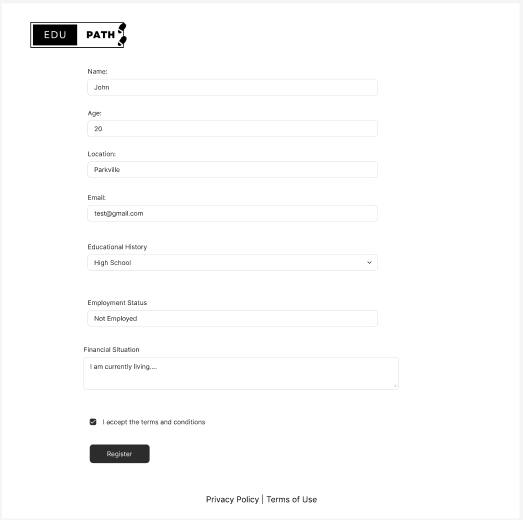
1. **Wireframes and Mockups (UI/UX Design)**

**Welcome and Registration Screen**

This is the first screen users see. The goal is to collect essential user information to personalize their experience.

Key Elements:

* Logo and App Name: EduPath at the top with a welcoming message.
* Input Fields:
* Name, Age, Location, Email.
* Dropdown for educational history (High school, VET, etc.).
* Employment Status and Financial Situation.
* A checkbox for accepting the terms and conditions.
* Call to Action: A Register or Get Started button.
* Footer: Link to privacy policy and app terms.

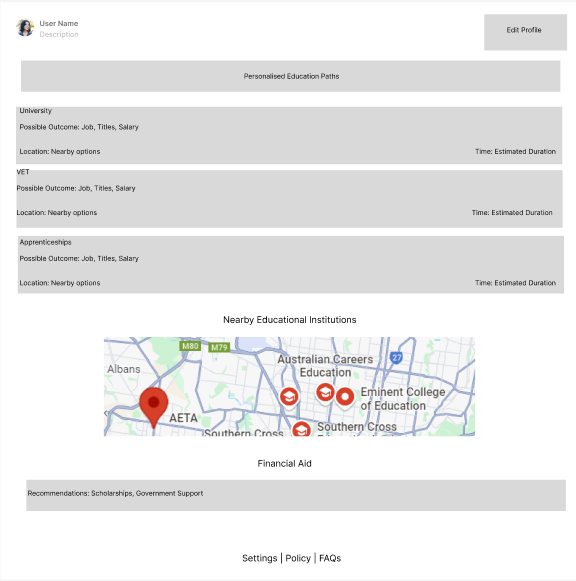
****

**2. User Dashboard**

**Once the user registers, they land on their dashboard, where they see personalized recommendations based on their profile.**

**Key Elements:**

* Header: User’s name, a profile picture, and a quick Edit Profile link.
* Main Sections:
* Personalized Education Paths:
* Cards showing different options like University, VET, and Apprenticeships.
* Each card should display basic info like possible outcomes (job titles, salary range), location, and the estimated time to complete.
* Map: A small, interactive map showing nearby educational institutions and training centers.
* Financial Aid: A section with recommendations for scholarships, government support, or financial aid packages.
* Footer: Quick links to settings, policy information, and FAQs.

****

**3. Recommendations Screen**

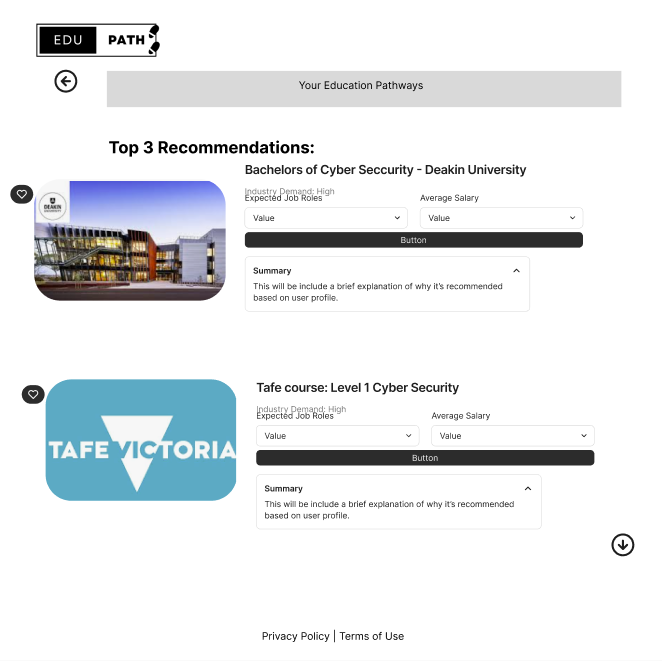
This screen provides detailed, personalized education and career recommendations for the user based on their profile and inputs.

Key Elements:

* Header: A back button to return to the dashboard and a clear title like “Your Education Pathways.”
* Main Content:
* Top 3 Recommendations: Display three top education/training paths (University, VET, Apprenticeship), each showing:
* Title of the Program
* Summary: A brief explanation of why it's recommended based on the user's profile.
* Outcomes: Expected job roles, average salary, industry demand.
* Duration: Estimated time to complete.
* Call to Action: A button to explore more about each option.
* More Options: Below the top 3, include a scrollable list of additional education options, possibly with filters by location, cost, or duration.

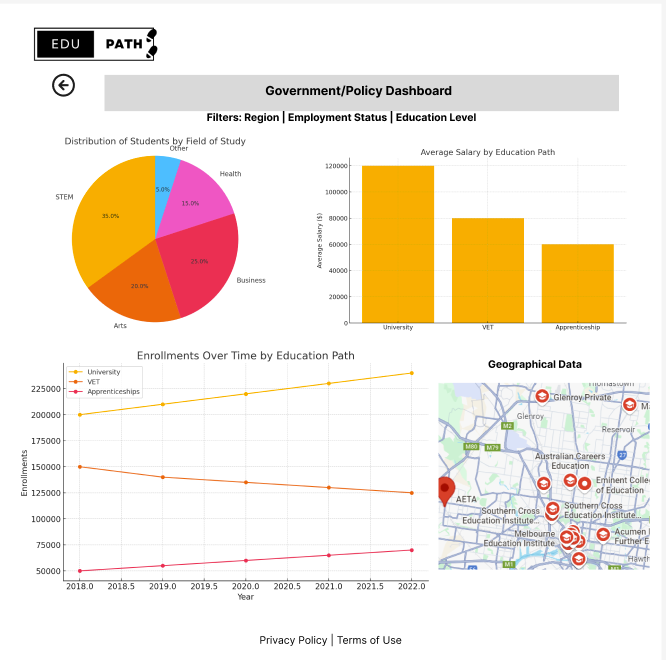
**Input Forms**: Show how users can input their information (age, location, employment status, financial situation, education history).

* **Recommendations Screen**: Display educational paths based on user input (University, VET, Apprenticeships) with details about each option.
* **Insights and Reports Section**: Show how government agencies and career advisors would view data-driven insights and trends.



**5. Government/Policy Dashboard**

* **Create Data Visualizations**:
  + Use Figma's built-in shapes (circles, bars) to create mockups of pie charts, bar graphs, or line charts.
  + Add filters at the top to simulate how a government official might filter the data by region, employment status, etc.
* **Add a Map for Geographical Data**:
  + Place a map with color-coded regions showing education accessibility.



**User Flow:**

* **Step 1**: User inputs personal data (age, location, employment, education, etc.).
* **Step 2**: The system processes the data using the AI model and provides education/training options.
* **Step 3**: Users can explore financial aid and employment opportunities alongside educational options.
* **Step 4**: Government/career advisors can access aggregated data on user trends and decision-making patterns.

**AI Model for EduPath**

**Type of Model:**

* **Supervised Learning Model**: Since the goal is to predict educational and employment outcomes based on personal data, a **supervised learning model** would be the most suitable.
* **Hybrid Model (Classification and Regression)**:
  + **Classification**: To predict the type of education path (University, VET, Apprenticeship).
  + **Regression**: To predict outcomes such as future income, employment stability, or likelihood of financial aid eligibility.

**Model Options:**

* **Random Forest Classifier/Regressor**: Can handle both categorical and continuous data (ideal for classification tasks like predicting education choices and regression tasks like predicting income).
* **Logistic Regression**: Simple and effective for binary or multi-class classification (e.g., predicting which educational path the user is most likely to choose).
* **Neural Networks** (Optional for more complex patterns): If the dataset is large, a neural network can help identify non-linear relationships between variables like financial status, proximity to institutions, and employment outcomes.

**Key Features for Training:**

1. **Age and Educational History**: Information such as the user’s highest level of education completed, current student status, etc.
2. **Employment Status and Financial Data**: Whether they are employed, part-time, full-time, unemployed, income level, access to scholarships or financial aid.
3. **Geographic Location**: Proximity to educational institutions, access to transportation, rural vs. urban location.
4. **Health and Disabilities**: Information that could impact their education decisions (e.g., need for flexible learning environments).

**Training Process:**

1. **Data Collection and Preprocessing**:
   * **Collect datasets** from sources like ABS, HILDA, and NCVER that provide education, employment, financial, and health data.
   * **Clean and normalize** the data (handle missing values, convert categorical features into numerical ones, etc.).
   * **Split Data**: Train-test split (e.g., 80% training, 20% testing) to evaluate the model's performance.
2. **Feature Selection**:
   * Select key features like education level, employment status, age, location, and financial aid status.
3. **Train the Model**:
   * Use selected algorithms (Random Forest, Logistic Regression, etc.) to train the AI model.
   * **Evaluate Performance**: Use metrics like **accuracy**, **precision**, **recall** (for classification) and **Mean Squared Error (MSE)** or **Root Mean Squared Error (RMSE)** for regression tasks.
4. **Model Deployment**:
   * Once trained and evaluated, the AI model can be integrated into the back-end of the application to make real-time predictions based on user input.

**Application Architecture**

**Front-End Development:**

The front-end will focus on user interaction, displaying data, and gathering inputs from users.

* **Programming Languages**:
  + **JavaScript**: The core programming language for building interactive elements in the web application.
  + **React.js**: A **JavaScript library** for building user interfaces. React is ideal because it allows component-based development, which is highly scalable and easy to maintain.
  + **HTML/CSS**: For building the structure and styling the application. This ensures that the app has a clean, modern interface.
* **Key Front-End Features**:
  + **User Dashboard**: Interactive cards that display education options, financial aid, and a map showing nearby institutions.
  + **Forms**: Used for collecting user input (age, location, employment status, financial situation, etc.).
  + **Data Visualization**: Charts and graphs for government and users, showing trends in education and employment choices.

**Back-End Development:**

The back-end will handle the core functionality, including processing user data, running the AI model, and managing the database.

* **Programming Languages**:
  + **Python**: Ideal for back-end development, especially when working with machine learning models. Frameworks like **Flask** or **Django** can be used to build the API that connects the AI model with the front end.
  + **Node.js**: Another option for the back-end, especially for real-time data handling. It works well with JavaScript-based front-end frameworks like React.js.
* **Framework**:
  + **Flask/Django** (for Python): Flask is a lightweight option for creating REST APIs and integrating with AI models. Django is a more comprehensive framework that comes with built-in features like authentication and database management.
  + **Express.js** (for Node.js): If using Node.js for the back-end, Express.js would be a great framework for handling routing, managing requests, and serving the front-end.
* **Database**:
  + **PostgreSQL/MySQL**: A relational database is ideal for storing user information, education options, and financial data.
  + **MongoDB**: If you plan to store more flexible, unstructured data (e.g., user inputs, logs), MongoDB can be used as a NoSQL database.

**Integration and Workflow**

Here’s how the front-end, back-end, and AI model would work together:

1. **User Interaction**:
   * The user inputs their personal information (age, location, education, employment, etc.) into the front-end form (built using React.js).
2. **Data Flow to Back-End**:
   * The front-end sends the data to the back-end API (built using Flask/Django or Node.js).
3. **AI Model Processing**:
   * The back-end processes the data and passes it to the AI model for prediction (e.g., predicting the most suitable education/training path and future employment outcomes).
4. **Return Results**:
   * The AI model returns the results (e.g., top 3 education paths, expected salary) to the back-end.
   * The back-end sends the processed data back to the front-end, where it is displayed to the user on the dashboard.

**Deployment**

Once developed, the app would need to be deployed on cloud platforms such as:

* **Amazon Web Services (AWS)** or **Google Cloud Platform (GCP)**: These platforms offer services for deploying both the front-end and back-end, hosting the AI model, and managing databases.
* **Heroku**: For easier deployment, especially for prototyping and lightweight apps.
* **Docker**: Containers can be used for easy deployment of the front-end, back-end, and AI model as a single package.