MONASH UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

FIT5120 2025 INDUSTRY EXPERIENCE STUDIO PROJECT

DATA MANAGEMENT PLAN

Table of Contents

1. Data Sources and Update Processes

1.1 Open Data Sources

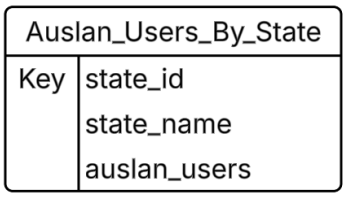
The following open datasets are used in the project:

Dataset Name	Physical Access	Frequency of Source Updates	Granularity	Link	Copyright	Iteration used
Auslan Sign Language Fingerspelling	Images	Never	Per frame/hand position	https://www.kaggle.com/datasets/rithwikchhugani/auslan-sign-language-fingerspelling-dataset/data	CC0 1.0 Universal	N/A
Auslan Daily Dataset	Images/Videos	Never	Per person/Per action	https://uq-cvlab.github.io/Auslan-Daily-Dataset/docs/en/dataset-download	Creative Commons BY 4.0 license.	N/A
National Health Survey	XLSX	Yearly	Per respondent	https://www.abs.gov.au/statistics/health/health-conditions-and-risks/national-health-survey/2022#data-downloads	ABS Copyright	1
NDIS PB Hearing Data	XLSX	Annually	Per region/ participant group	https://dataresearch.ndis.gov.au/reports-and-analyses/participant-dashboards/hearing-impairment	Creative Commons BY 4.0 license.	1
Burden of disease broad disease groups	CSV	Yearly	Per disease group	https://www.data.act.gov.au/Health/Burden-of-disease-broad-disease-groups/jv94-5gan/about_data	CC0 1.0 Universal	1
Global Overview of hearing loss data	CSV	Yearly	Global/Region projection per year	https://www.statista.com/statistics/888569/number-of-people-with-hearing-loss-worldwide-projections/	CC0 1.0 Universal	1
Cultural diversity: Census	CSV	Never	Per demographic group	https://www.abs.gov.au/statistics/people/people-and-communities/cultural-diversity-census/latest-release#data-downloads	ABS Copyright	N/A

Word Validation	Txt/CSV	Never	Per word	https://www.kaggle.com/datasets/bwandowando/479k-english-words?resource=download	CC0 1.0 Universal	2
Auslan Signs	Images	Never	Per frame	https://drive.google.com/drive/folders/1HO9E143LEVOyjuq6uU9oHG64uGrZ6Xu3	CC0 1.0 Universal	3

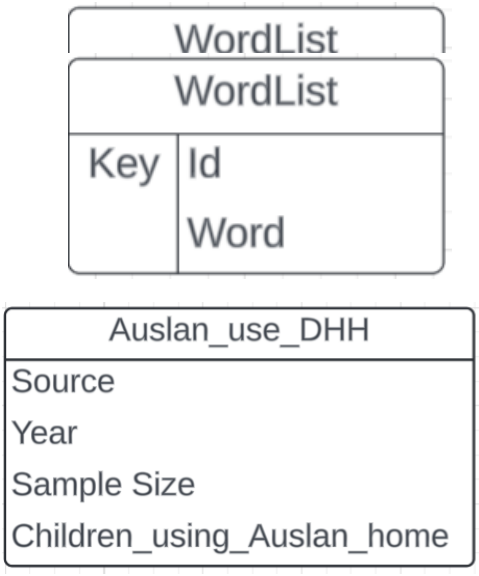
2. Data Modelling and Analysis

2.1 ER Diagram



Iteration 1

Iteration 2



Iteration 3

3. Data Wrangling and Cleansing

This section explains the processes followed for cleaning and preparing two datasets for analysis and visualization: *PB Hearing data to 30 June 2023* and the *Cultural Diversity data summary*.

3.1 Data Wrangling

Data Wrangling – PB Hearing Data *Iteration 1*

Data wrangling involved reformatting and reshaping the raw data into an analysis-friendly structure. The critical steps were:

- Sheet Selection: Among the 33 sheets of the Excel file, 24_Outcome_chldrn was selected as it has a direct impact on children's outcomes with hearing impairment.
- Header Adjustment: The initial row of the dataset was a descriptive header and was not part of the data. It was removed to enable correct column name alignment.
- Column Renaming: Column names were renamed for greater meaningfulness:
    - First Reassessment
    - Latest Reassessment
    - Percent Movement
    - Outcome (description of the impact being measured)
- Column Removal: A placeholder unnamed column was discovered and removed as it did not contribute anything to the analysis.
- Index Reset: Once the columns and top row were cleaned, the index was reset to provide a constant referencing.

Data Wrangling - Cultural Diversity Data Summary *Iteration 1*

This part outlines the processing of Table 5 from the *Cultural Diversity data summary*, specifically focused on extracting data about Auslan (Australian Sign Language) use across states and territories. The critical steps include:

Table Selection: Table 5, titled *"LANGUAGE USED AT HOME BY STATE AND TERRITORY"*, was selected as it contains language use data across all Australian states and territories.

Target Row Identification: The row labeled **"Auslan"** was located, which reflects the number of users of Australian Sign Language.
Column Filtering: Only the following columns were retained for analysis:

- New South Wales
- Victoria
- Queensland
- South Australia
- Western Australia

- Tasmania
- Northern Territory
- Australian Capital Territory

Numeric Type Enforcement: All values were cast to integer type to ensure compatibility with later aggregation and analysis.

No Wrangling done for Iteration 2 and 3 as just cleaning is required.

3.2 Data Cleansing

Data Cleansing – PB Hearing Data *Iteration 1*

Data cleaning was performed to reach accuracy, consistency, and usability of the dataset. The data cleaning process involved:

- Data Type Conversion: The percentage values under First Reassessment, Latest Reassessment, and Percent Movement were explicitly converted into float type so that numerical analysis and visualization would be possible.
- Missing Value Handling: There were no significant missing values in this specific sheet. Checks were however performed using pd.to_numeric(., errors='coerce') to handle any non-numeric inputs in a graceful way.
- Export for Reusability: The cleaned data was exported as cleaned_outcome_children.csv for ease of reuse in several steps of visualization and reporting.

Data Cleansing – Cultural Diversity Data Summary *Iteration 1*

- **Header Normalization**: Column headers and contents were stripped of any leading/trailing whitespace or formatting inconsistencies.

- **Missing Value Check**: The selected row for Auslan had complete data across all regions.

- **Validation**: The sum of the state-level values was cross-checked against the "Total" value (16,242) to verify consistency and data integrity.

- **Export**: The final cleaned data was saved as `auslan_by_state.csv` for downstream usage in dashboards or reports.

Data cleaning - Word validation: *Iteration 2*

```
def clean_dataframe(df):
    # 1. Remove duplicates
    df = df.drop_duplicates()

    # 2. Remove empty strings
    df = df[df['word'].str.strip() != '']

    # 3. Strip extra spaces just in case
    df['word'] = df['word'].str.strip()

    return df
```

This function is used to clean the raw word list extracted from a text file. It performs the following critical data wrangling tasks:

- **Duplicate Removal**: Keeps the dataset with only unique entries, which is required for further processing or analysis that assumes uniqueness.
- **Empty String Filtering**: Eliminates empty or invalid entries that may have been inserted due to formatting issues or line breaks.
- **Whitespace Trimming**: Cleans entries by removing leading or trailing whitespace that might interfere with matching or indexing.

These operations are essential to the maintenance of data quality, consistency, and reliability, which directly impacts the correctness of any ensuing database indexing.

Iteration 3 - No Data Cleansing as we are using images for the model.

4. Code Snippet on Core Functionalities

Iteration 2

Persistent Learning Progress Using Local Storage

To create an engaging, personalized experience for our target users—primary school children—we designed the SignMates application to support learning continuity without the need for user accounts. This was achieved through the use of the browser's localStorage, which allows us to persist key user interactions, particularly which signs the user has learned. This approach ensures that learners can resume their progress across visits and receive personalized feedback during gameplay, all without the friction of registration or login.

Saving and Retrieving Learnt Signs

In the **Learn Page**, each sign is accompanied by a checkbox labeled "I know this sign." When the user ticks the box, the app stores that sign's identifier in the appropriate category (letter, number, or common) within a learntSigns object. This object is then serialized to JSON and saved into localStorage. On subsequent visits, this data is retrieved, parsed, and used to update the app's internal state so the progress bar and checkbox status reflect the user's previous interactions.

Code snippet: Saving progress

```
const toggleLearnt = (category, key) => {
    const updated = {
        ...learntSigns,
        [category]: learntSigns[category].includes(key)
            ? learntSigns[category].filter(k => k !== key)
            : [...learntSigns[category], key],
    };
    setLearntSigns(updated);
    localStorage.setItem('learntSigns', JSON.stringify(updated));
};
```

Code snippet: Loading progress on page load

```
useEffect(() => {
    const stored = localStorage.getItem('learntSigns');
    if (stored) {
        try {
            setLearntSigns(JSON.parse(stored));
        } catch (err) {
            console.error('Failed to parse learntSigns from localStorage:', err);
        }
    }
}, []);
```

By storing progress data locally, users benefit from a lightweight, frictionless learning experience. This method is particularly suitable for children, who may not have email

addresses or accounts to log in with. It also avoids introducing privacy concerns or data collection issues.

Adapting Gameplay Based on Learning Progress

The learntSigns data is not only used in the Learn Page—it also informs gameplay across our suite of mini-games, particularly in the Memory Match game. When starting a game, users are offered two gameplay modes:

- All Signs Mode: where the full set of common signs is used, regardless of whether the user has learned them.
- Learnt Mode: a specialized game mode that limits cards to only the signs the user has previously marked as learned.

This adaptive feature adds pedagogical value by reinforcing prior learning while maintaining engagement. To ensure meaningful gameplay, Learnt Mode is only unlocked when the user has marked at least four common signs as learned. If fewer than four signs have been learned, this mode is visually disabled and accompanied by a message encouraging the user to continue learning.

Code snippet: Conditional access to Learnt Mode

```
<label className={learntSigns.length < 4 ? 'disabled' : ''}>
    <input
        type="radio"
        name="signMode"
        value="learnt"
        disabled={learntSigns.length < 4}
        checked={signMode === 'learnt'}
        onChange={() => setSignMode('learnt')}
    />
    Learnt signs

    {learntSigns.length < 4 ? (
        <div className="option-description disabled">
            🧠 You've learnt <strong>{learntSigns.length} common signs</strong> so far! <br />
            📖 Keep learning – unlock this mode after learning at least <strong>4 common signs</strong>.<br />
            👉 <button onClick={() => navigate('/learn')}>Go to Learn</button>
        </div>
    ) : (
        <div className="option-description">
            ✅ You've learnt <strong>{learntSigns.length} signs</strong>! <br />
            🎯 Pick this mode to practice signs you already know. <br />
            👉 <button onClick={() => navigate('/learn')}>Want to learn more?</button>
        </div>
    )}
</label>
```

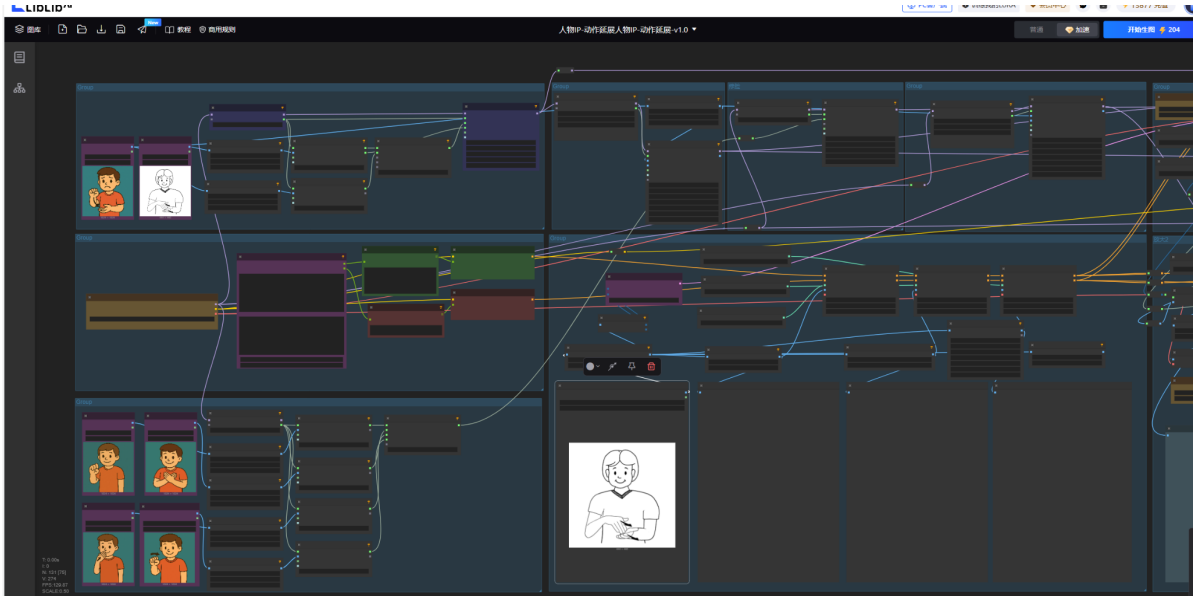
In addition, the gameplay logic uses the same learntSigns object to determine which signs are included in the game. When Learnt Mode is selected, only the filtered set of learned signs is used for the memory card pairs:

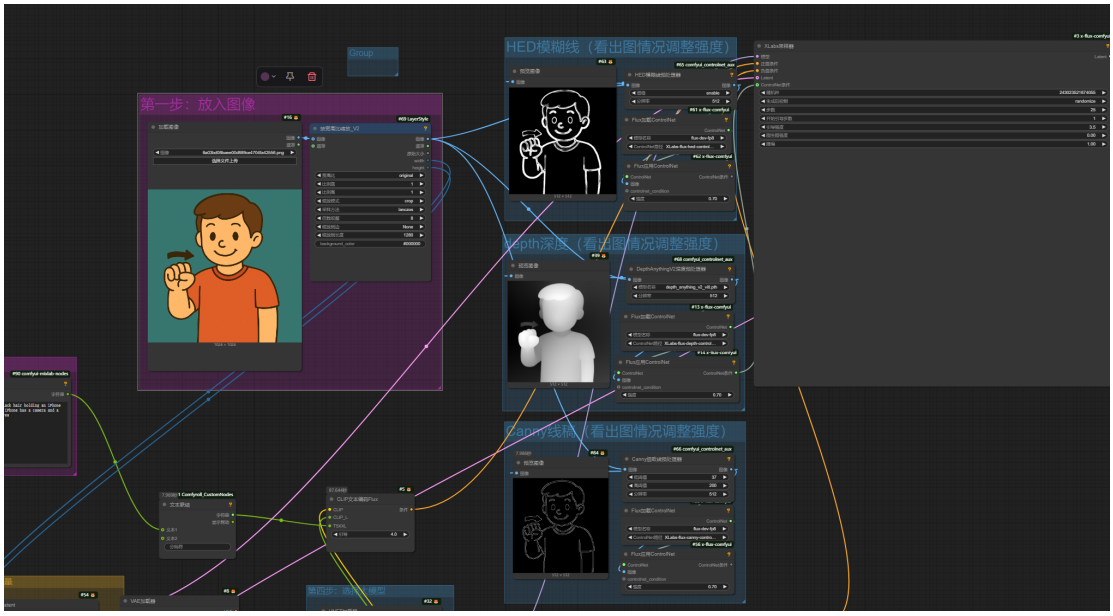
Code snippet: Filtering game content based on learnt signs

```
const handleStartGame = () => {
    const selectedSignSet =
        signMode === 'learnt'
            ? Object.fromEntries(
                learntSigns.map(key => [key, allSigns[key]]).filter(([_, val]) => val)
            )
            : allSigns;

    const limitedKeys = shuffleArray(Object.keys(selectedSignSet)).slice(0, 4);
    const limitedSigns = {};
    limitedKeys.forEach(key => {
        limitedSigns[key] = selectedSignSet[key];
    });

    setSigns(limitedSigns);
    setShuffledCommonKeys(shuffleArray(limitedKeys));
    setShuffledSignEntries(shuffleArray(Object.entries(limitedSigns)));
    setIsGameStarted(true);
    if (firstVisit)
        setShowInstructions(true);
};
```

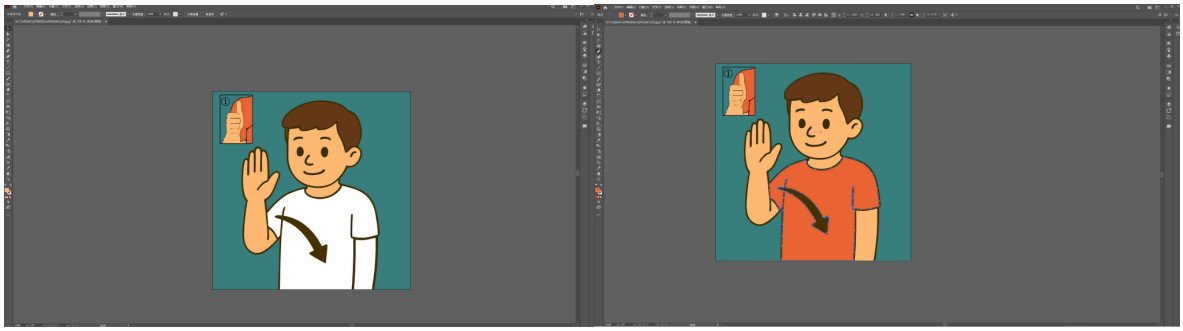
Lexical Words images creation:

Lexical word images were drawn using stable diffusion mapping software. I used ComfyUI as a node-based visual editing tool to build a flowchart containing text prompts (Prompt), gesture control and image generation. By introducing the OpenPose module, key skeletal points for sign language movements are fed into ControlNet to precisely control the character's gestures, direction of movement, and pose structure, here is the screenshot of the stable diffusion:





Also, I use Photoshop to modify the images I created:



Educational and Design Benefits

This use of localStorage across both the Learn and Game pages enables a seamless and consistent user experience. The benefits include:

- **Progress Continuity:** Users can leave the site and return later without losing track of their learning.
- **Adaptive Gameplay:** Games dynamically adjust to the user's learning level, making them more effective and rewarding.
- **No Sign-In Barrier:** The app avoids requiring user registration, aligning with the accessibility needs of young learners.
- **Visual Motivation:** The progress bar and the unlocking of game modes serve as motivational tools, encouraging continued engagement.

By leveraging localStorage, we struck a balance between technical simplicity and educational value. This implementation supports autonomy and intrinsic motivation, especially in a school setting where children may share devices or revisit the tool over multiple sessions. It also aligns with broader project goals of being low-barrier, privacy-conscious, and age-appropriate.

Iteration 3

Game logic and Educational content mapping:

"The achievement feedback and conversation system are incorporated through controlled code and metadata mappings to ensure modularity, consistency, and access within the learning game environment."

```
// Choose emoji based on achievement text
let achievementEmoji;
if (achievementText.includes('Knowledge Badge')) {
  achievementEmoji = '🏆';
} else if (achievementText.includes('Hearing Aid')) {
  achievementEmoji = '🦻';
} else {
  achievementEmoji = '📚';
}
ctx.fillText(achievementEmoji, boxX + 30, boxY + 45);
```

This code snippet dynamically maps achievement descriptions to specific emojis based on keywords in the achievementText. It is applied in the visual feedback system of the game for player achievements. The selection of emojis enhances user engagement and accessibility in terms of visual representation of earned badges, such as:

- for overall academic achievement ("Knowledge Badge")
- for deaf and hard-of-hearing (DHH) awareness achievements ("Hearing Aid")
- as an alternate education symbol

This data-based conditional logic validates the display and makes it applicable and responsive to what the player is doing. It uses CanvasRenderingContext2D.fillText() to display the selected emoji on the interface.

```
// Dialogue sequences for each NPC
export const NPC_DIALOGUES = {
  [NPC_IDS.TEACHER1]: {
    initial: [
      { speaker: "Class Teacher", text: "Welcome! I'm glad you're here. We have a new student joining our school today - Alex, who is Deaf and Hard of Hearing
      { speaker: "Class Teacher", text: "Let me tell you about DHH individuals and their experiences. Did you know that DHH people face unique challenges in c
      {
        speaker: "Class Teacher",
        text: "Question: What is the primary challenge faced by DHH individuals in a hearing world?\n1. Physical mobility\n2. Communication barriers\n3. Visua
        isQuestion: true,
        correctAnswer: 2
      },
      { speaker: "Class Teacher", text: "Communication barriers are the primary challenge. DHH individuals often face difficulties in understanding spoken lang
      { speaker: "Class Teacher", text: "Please see the Language Teacher after this conversation to learn more about how we can communicate with DHH individua
      { speaker: "Class Teacher", text: "Language Teacher is waiting for you near the Library." }
    ]
  },
```

This statement declares structured dialogue data for an NPC, here a teacher that has DHH related issues to show to the player. The dialogue is employed as an array of consecutive text interactions, allowing both narrative speech and interactive query capabilities.

Key features:

- speaker and text fields hold the scripted dialogue.
- isQuestion: true flag identifies quiz interactions.
- correctAnswer: 2 states the correct option, after accessible content on DHH challenges.

This design supports the delivery of educational content within the game flow in addition to tracking user understanding. Dialogue chains are done as constants and accessed by character ID for easy scalability and localization.

5. Database Security & Backup Plan

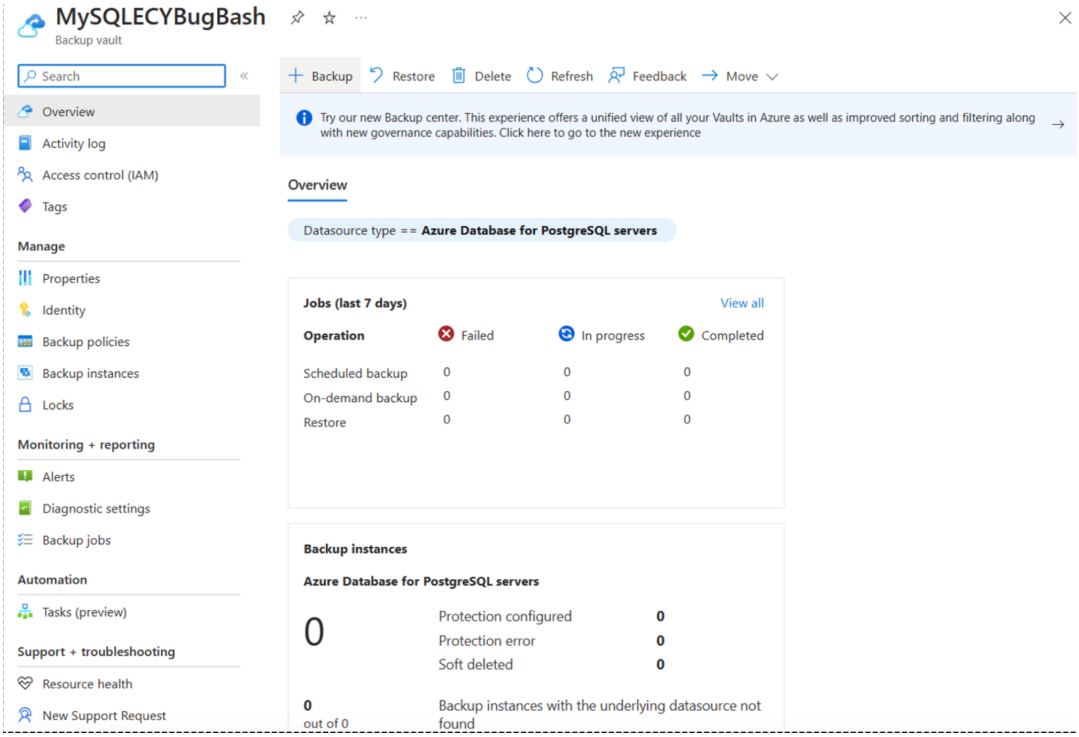
Database backup strategy uses custom encrypted full logical backups using `mysqldump` and Azure Blob Storage (Microsoft Azure, 2024). This approach provides reliable disaster recovery and portable, auditable backups stored securely in the Azure cloud, in line with best practices from MySQL (Oracle, 2024) and Microsoft documentation (2024).

5.1 Backup Process

Full Backups via `mysqldump`

In addition to Azure's built-in automatic backup, we generate **weekly full logical backups** using the open-source `mysqldump` utility:

- Backups are scheduled using **cron jobs** or **Azure Automation Runbooks** on a secure virtual machine.
- Resulting `.sql.gz` files are **encrypted with AES-256** using tools like `gpg` before upload.
- Encrypted backup files are uploaded to **Azure Blob Storage** using **Azure CLI** or Python SDK.
- Storage containers are protected using **Shared Access Signatures (SAS tokens)** and limited access policies.
- Upload destinations follow naming conventions like:
  `azure://backup-db/mydb-YYYY-MM-DD.sql.gz`



Automated Backups via Azure

Azure Database for MySQL includes:

- **Daily snapshots** with up to **35 days of retention**.
- **Point-in-Time Restore (PITR)** to any moment within the backup window.
- Storage in **geo-redundant Azure Storage (GRS)** for disaster tolerance.
- Backup encryption managed by **Microsoft or customer-provided keys (CMK)** via Azure Key Vault.

5.2 Retention & Lifecycle

We implement a tiered backup retention policy based on usage needs and compliance considerations.

Backup Type	Frequency	Retention	Storage Tier	Encryption	Access
Azure Automatic Backup	Daily	7-35 Days	GRS (built in)	Microsoft Managed	RBAC +AAD

Logical Backup via mysqldump	Weekly	90 Days	Azure Blob (Hot – Archive)	AES 256	SAS Token + ACL

- After **30 days**, logical backups are automatically moved to the **cool tier**, and to **archive** after 60 days using **Azure Blob Lifecycle Management**.
- Archived backups are restored within hours, supporting disaster recovery and audits.
- Blob versioning and **immutability policies** are optionally enabled to meet compliance (e.g., ISO 27001, HIPAA).

5.3 Disaster Recover and Testing

To ensure the reliability of our backup strategy, we conduct regular disaster recovery drills on a quarterly basis. These tests involve restoring encrypted `.sql.gz` backups into a separate staging environment, followed by checksum validation to verify data integrity. The entire restoration process is documented in a formal Disaster Recovery Runbook, which outlines roles, procedures, and rollback steps in case of critical failures. Our objective is to maintain a Recovery Point Objective (RPO) of 24 hours and a Recovery Time Objective (RTO) of under one hour, ensuring that any unexpected data loss or corruption can be addressed quickly and confidently.

6. Ethical, Legal, and Privacy Issues

6.1 Ethical Considerations

Data Bias:

- **Demographic Representation:** Ensure the datasets used do not over-represent specific demographics. For instance, the Auslan Sign Language Fingerspelling Dataset and the Auslan Daily Dataset should be verified to contain a wide range of signers across various age groups, genders, and cultures.
- **Health Data:** When dealing with health-related data sets like the National Health Survey, 2022 and the Burden of Disease data, it is essential to regularly audit for biases so that outcomes are not skewed towards certain populations.

Transparency:

- **Documentation:** Clearly document all data sources and processing steps. For example, when combining data from the NDIS Hearing Impairment reports, clearly describe how the data was obtained, processed, and interpreted.

- **User Information:** Inform users about the use of their information, especially if any information is collected directly from them. Transparency establishes trust and ensures ethics compliance.

6.2 Legal and Privacy Issues

Data Licensing:

- **Open-Source Compliance:** Ensure that all data sets are open-source and used according to their licenses. For instance:
    - The Auslan Sign Language Fingerspelling Dataset is available on Kaggle; review its terms of licensing before use.
    - The Auslan Daily Dataset is licensed under Creative Commons BY 4.0, which permits sharing and adaptation with attribution.

Privacy:

- **Anonymity:** Ensure that no personally identifiable information (PII) is stored or processed. Datasets like the National Health Survey, 2022 provide aggregated data, which minimizes privacy concerns.
- **Data Aggregation:** When dealing with location-based data, such as UV exposure data, ensure that data is aggregated so that individuals' exact locations are not revealed.
- **No Personal Information Collected:** We do not collect, store, or share any personal information from users.
- **No Tracking:** We do not use cookies, advertising tools, analytics services, or third-party tracking technologies on this website.
- **Local Progress Storage:** Any learning progress, such as signs marked as learned, is saved only on your device using your browser’s local storage. This information is not sent or accessible to us.
- **No Account or Login:** You are not required to create an account or provide any personal details to use SignMates. All features are accessible without registration.
- **External Links:** Some pages may link to external educational resources. These sites have their own privacy policies. Please review them if you choose to visit those links.
- **Children’s Privacy:** SignMates is designed to be safe for children. We do not collect or store any data from children or identifiable information.

7. Future Needs and Challenges

7.1 Future Needs

Scalability:

System Expansion: Design the system architecture to support increasing capacities of data when additional datasets and postcodes are combined. This is particularly required when combining large datasets like the National Health Survey.

Real-time Updates:

API Performance: Make API calls efficient to support real-time data retrieval and processing while upholding timely updates for users, especially during the combination of dynamic datasets.

7.2 Potential Challenges

Data Quality
Consistency: Maintain uniform data quality across all integrated datasets. Regularly validate and clean data from sources like the Burden of Disease dataset to avoid untrustworthiness.

API Limitations
Rate Limits and Downtime: Implement mechanisms to manage API rate limits and inevitable downtimes in a graceful way, ensuring uninterrupted data availability and user experience.

8. Conclusion

This Data Management Plan defines the application's practices in sourcing, updating, and data management. In addressing ethical, legal, and privacy issues, and exemplifying vast knowledge in data modeling and data analysis practices, the plan is guaranteed for sound and effective use of data. Regular review and updating will happen in every release to maintain alignment with project goals and current data practices.