```
                    ┌─────────────────────┐
                    │      Input:         │
                    │ Alzheimer's data (gene │
                    │ expression/EEG/ MRI)  │
                    └─────────────────────┘
                              │
                              ▼
        ┌──────────────────────────────────────────┐
        │                                          │
        │              ╲   Encoder   ╱             │
        │               ╲_____╱              │
        │                    │                     │
        │                    ▼                     │
        │  adVAE      ┌──────────────┐             │
        │  model      │ Latent Variables │         │
        │             └──────────────┘             │
        │                    │                     │
        │                    ▼                     │
        │               ╱‾‾‾‾‾‾‾‾‾‾‾╲              │
        │              ╱   Decoder   ╲             │
        │                                          │
        └──────────────────────────────────────────┘
                              │
                              ▼
                    ┌─────────────────────┐
                    │      Output:        │
                    │ synthetic Alzheimer's │
                    │        data         │
                    └─────────────────────┘
```

Model Architecture

Functional Description

adVAE aims to:

1. Aid Alzheimer's research by creating high-quality multimodal synthetic data (gene expression, EEG, MRI)

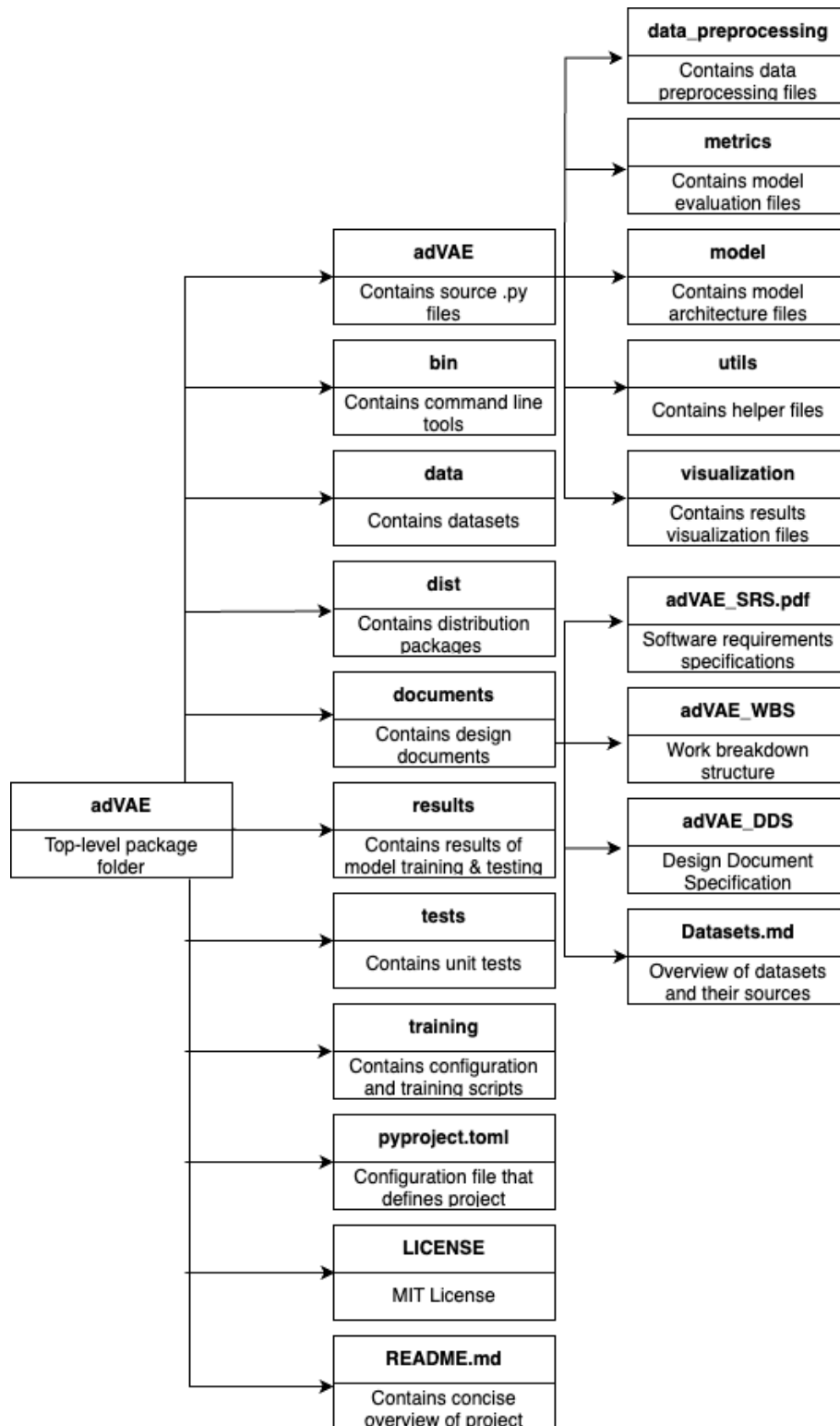2. Easily integrate with existing biomedical workflows

User Interface

Command line interface (CLI)

Functionality

1. Preprocess input data used to train the VAE
2. Latent space representation of various data modalities
3. Generation of synthetic data using latent space variables

Task Prioritization

1. Identify example data (gene expression, EEG and MRI)

2. Create VAE model architecture for one modality (gene expression)

3. Train the model using the example gene expression dataset

4. Perform latent space optimization

5. Validate the results by calculating KL loss and reconstruction error of generated dataset

6. Extend the model to include other modalities

7. Train, validate, and test the model using the bulk data

Goals and Milestones

1. Finish all design documentation

2. Set up GitHub repository

3. Create modules

4. Start implementing code

5. Create model for one modality

6. Extend model for other modalities

7. Peer review

8. Refine model and update design documentation

Solutions

1. Tensorize all input data

2. Create separate modules/classes for handling different data modalities

3. Validate by calculating KL loss and reconstruction loss

DDS

```
                                          ┌─────────────────────────┐
                                          │   data_preprocessing    │
                                          ├─────────────────────────┤
                                          │      Contains data      │
                                          │   preprocessing files   │
                                          └─────────────────────────┘

                                          ┌─────────────────────────┐
                                          │         metrics         │
                                          ├─────────────────────────┤
                                          │     Contains model      │
                                          │    evaluation files     │
                                          └─────────────────────────┘

                ┌──────────────────┐      ┌─────────────────────────┐
                │      adVAE        │      │         model           │
                ├──────────────────┤      ├─────────────────────────┤
                │ Contains source .py│    │     Contains model      │
                │      files        │      │   architecture files    │
                └──────────────────┘      └─────────────────────────┘

                ┌──────────────────┐      ┌─────────────────────────┐
                │       bin         │      │         utils           │
                ├──────────────────┤      ├─────────────────────────┤
                │ Contains command line│   │   Contains helper files │
                │      tools        │      └─────────────────────────┘
                └──────────────────┘

                ┌──────────────────┐      ┌─────────────────────────┐
                │      data         │      │     visualization       │
                ├──────────────────┤      ├─────────────────────────┤
                │ Contains datasets │      │    Contains results     │
                └──────────────────┘      │   visualization files   │
                                          └─────────────────────────┘

                ┌──────────────────┐      ┌─────────────────────────┐
                │       dist        │      │     adVAE_SRS.pdf       │
                ├──────────────────┤      ├─────────────────────────┤
                │ Contains distribution│   │ Software requirements   │
                │     packages      │      │    specifications       │
                └──────────────────┘      └─────────────────────────┘

                ┌──────────────────┐      ┌─────────────────────────┐
                │    documents      │      │      adVAE_WBS          │
                ├──────────────────┤      ├─────────────────────────┤
                │ Contains design   │      │   Work breakdown        │
                │   documents       │      │     structure           │
                └──────────────────┘      └─────────────────────────┘

  ┌──────────────┐ ┌──────────────┐      ┌─────────────────────────┐
  │   adVAE      │ │    results   │      │      adVAE_DDS          │
  ├──────────────┤ ├──────────────┤      ├─────────────────────────┤
  │ Top-level    │ │ Contains results of│ │  Design Document        │
  │ package      │ │ model training & testing│ │  Specification      │
  │ folder       │ └──────────────┘      └─────────────────────────┘
  └──────────────┘
                ┌──────────────────┐      ┌─────────────────────────┐
                │      tests        │      │      Datasets.md        │
                ├──────────────────┤      ├─────────────────────────┤
                │ Contains unit tests│     │  Overview of datasets   │
                └──────────────────┘      │   and their sources     │
                                          └─────────────────────────┘

                ┌──────────────────┐
                │     training      │
                ├──────────────────┤
                │ Contains configuration│
                │ and training scripts│
                └──────────────────┘

                ┌──────────────────┐
                │   pyproject.toml  │
                ├──────────────────┤
                │ Configuration file that│
                │   defines project │
                └──────────────────┘

                ┌──────────────────┐
                │     LICENSE       │
                ├──────────────────┤
                │   MIT License     │
                └──────────────────┘

                ┌──────────────────┐
                │    README.md      │
                ├──────────────────┤
                │ Contains concise  │
                │ overview of project│
                └──────────────────┘
```

Module Structure

Tasks	Week 1 (02.03 - 02.09)							Week 2 (02.10 - 02.016)							Week 3 (02.17 - 02.23)							Week 4							Week 5							Week 6							Week 7							Week 8						
	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su
Create SRS																																																								
Create Datasets.md																																																								
Create WBS																																																								
Create Timeline																																																								
Create DDS																																																								
Identify data sources																																																								
Validate data																																																								
Implement first function																																																								
Split dataset into example-training-validation-test data																																																								
Perform data preprocessing for gene expression data (normalization, scaling, handling missing values)																																																								
Perform exploratory statistical analysis on the example data																																																								
Visualize the example data																																																								
Construct the VAE architecture																																																								
Set up hyperparameter tuning, backpropagation and training functions																																																								
Set up loss functions and other validation metrices																																																								
Train and test model on example dataset																																																								
Evaluate reconstruction quality																																																								
Optimize latent space for gene expression VAE model																																																								
Adapt model for MRI data																																																								
Adapt model for EEG data																																																								
Validate using full training, validation and test data																																																								
Develop Command Line Interface for adVAE																																																								
Update all design documentation																																																								
Summarize key findings																																																								
Prepare for peer review																																																								
Present findings																																																								

Timeline