Whole Exome Sequencing on the Illumina Novaseq 6000 NGS Platofrm

| Clinical Indication: |
| --- |
| {d.dynamic.clinicalSummary.data} |

| {d.dynamic.result.data} |
| --- |

\*

Variant Interpretation & Clinical correlation:

{d.dynamic.interpretation.data}

| Variant Evidence | Gene Impact |
| --- | --- |
| Chromosome: {d.dynamic.variantInfo.data.rowData[0][1]}  Position:  {d.dynamic.variantInfo.data.rowData[0][2]}   | Allele | DP | % | | --- | --- | --- | | {d.dynamic.variantInfo.data.rowData[0][4]} |  |  | |  |  |  |   Genotype:  {d.dynamic.variantInfo.data.rowData[0][4]} | RefSeq Gnese 110, NCBI   | Gene:  {d.dynamic.variantInfo.data.rowData[0][2]} | Transcript:  {d.dynamic.variantInfo.data.rowData[0][2]} | | --- | --- | | Effect:  {d.dynamic.variantInfo.data.rowData[0][6]} | Protein: | | Exon: | Coding: | |

Based on the evidence, this variant {d.dynamic.variantInfo.data.rowData[0][4]}

is classified as likely to be {d.dynamic.variantInfo.data.rowData[0][6]}

variant

OMIM Phenotype

{d.dynamic.variantInfo.data.rowData[0][6]}

| Recommendations |
| --- |
| * {d.static.recommendation.data[0]} * {d.static.recommendation.data[1]} * {d.static.recommendation.data[2]} * {d.static.recommendation.data[3]} |

| Methodology: |
| --- |
| {d.static.test\_methodology.data} |

| Limitations/Disclaimer: |
| --- |
| {d.static.limitation.data}  {d.static.disclaimer.data} |

Variant Classification as per ACMG guidelines:

| Name | Description |
| --- | --- |
| {d.static.terms.data.rowData[0][0]} | {d.static.terms.data.rowData[0][1]} |
| {d.static.terms.data.rowData[1][0]} | {d.static.terms.data.rowData[1][1]} |

References:

1. {d.static.references.data[0]}
2. {d.static.references.data[1]}
3. {d.static.references.data[2]}
4. {d.static.references.data[3]}
5. {d.static.references.data[4]}
6. {d.static.references.data[5]}
7. {d.static.references.data[6]}
8. {d.static.references.data[7]}
9. {d.static.references.data[8]}
10. {d.static.references.data[9]}
11. {d.static.references.data[10]}
12. {d.static.references.data[11]}
13. {d.static.references.data[12]}
14. {d.static.references.data[13]}
15. {d.static.references.data[14]}
16. {d.static.references.data[15]}
17. {d.static.references.data[16]}
18. {d.static.references.data[17]}
19. {d.static.references.data[18]}
20. {d.static.references.data[19]}
21. {d.static.references.data[20]}
22. {d.static.references.data[21]}
23. {d.static.references.data[22]}
24. {d.static.references.data[23]}
25. {d.static.references.data[24]}
26. {d.static.references.data[25]}
27. {d.static.references.data[26]}
28. {d.static.references.data[27]}

Appendix 1: Sample Data and Statistics

|  |  |
| --- | --- |
|  |  |
|  |  |

Appendix 2: Coverage Summary

| Mean Depth | Percentage target base pairs covered |
| --- | --- |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Appendix 3: Coverage of Analyzed Genes (Percentage of coding region covered)

| Gene | Region Covered | Gene | Region Covered | Gene | Region Covered |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

Database Information:

{d.dynamic.databaseInformation[0].description}

|  | 1000 Genomes | Indigen | gnomAD\_exome | gnom\_genome | inhousedb | GME |
| --- | --- | --- | --- | --- | --- | --- |
| Presence/Absence | {d.dynamic.databaseInformation[0].data.rowData[0][0]} | {d.dynamic.databaseInformation[0].data.rowData[0][1]} | {d.dynamic.databaseInformation[0].data.rowData[0][2]} | {d.dynamic.databaseInformation[0].data.rowData[0][3]} | {d.dynamic.databaseInformation[0].data.rowData[0][4]} | {d.dynamic.databaseInformation[0].data.rowData[0][5]} |
| MAF Vaues | {d.dynamic.databaseInformation[0].data.rowData[1][0]} | {d.dynamic.databaseInformation[0].data.rowData[1][1]} | {d.dynamic.databaseInformation[0].data.rowData[1][2]} | {d.dynamic.databaseInformation[0].data.rowData[1][3]} | {d.dynamic.databaseInformation[0].data.rowData[1][4]} | {d.dynamic.databaseInformation[0].data.rowData[1][5]} |

Additional Information:

{d.dynamic.additionFindingsHeader.data[0]}

Django Framework:

* Backend handles the logic for converting JSON data into a PDF format,Frontend provides a user-friendly interface for uploading JSON and DOCX template files.
* Carbone API is used to authenticate requests to the Carbone service.
* Template Handling - handles docx template and json file with data.User can upload both

Workflow

Data Mapping: The JSON data is parsed and mapped onto the placeholders in the DOCX template.

* The mapping process in the project involves inserting JSON data into the placeholders within the DOCX template.
* When a user uploads a JSON file and a DOCX template, the Django backend parses the JSON data and identifies the corresponding placeholders in the DOCX file.
* These placeholders are typically denoted by specific tags or markers (e.g., {d.dynamic.caseMetaData.proband.name}).
* The system then replaces these placeholders with the actual data from the JSON file.   
    
  For instance, if the JSON contains a patient's name, the system will find the placeholder for the patient's name in the template and replace it with the actual name from the JSON data.
* This process is repeated for all relevant fields, ensuring that the final DOCX document is fully populated with the provided data.
* Once the template is populated, it is sent to the Carbone API for conversion to a PDF document, which can then be downloaded by the user.
* This dynamic mapping ensures that the generated PDF accurately reflects the data provided in the JSON file, adhering to the format specified in the DOCX template.
* Users access the Django application through a web browser.
* The interface allows users to upload a DOCX template and a JSON file.
* Data Mapping
* The populated DOCX template is sent to the Carbone API for conversion to PDF.
* The Carbone API processes the request and returns the PDF document
* The generated PDF is made available for download through the user interface.