American International University- Bangladesh (AIUB)

Department of Computer Science
Faculty of Science & Technology (FST)

Project: AMR Prediction Pro

A Software Quality and Testing Project Submitted
By

Semester: Fall_23_24				**Section:**	**Group No:**
SL	**SN**	**Student Name**	**Student ID**	Individual Contribution (in %)	Total Marks: 50 Earned Marks:
A	2	S. S. M. AFSAN SANI	19-39332-1		
B	4	SARJAT AZIZ RUMI	19-40595-1		
C	15	FABLIHA HASNIN DUTI	21-44396-1		
D					

The project will be Evaluated for the following Course Outcomes

EVALUATION CRITERIA	**Total Marks (50)**
System Features	[10 Marks] **A:** **B:** **C:** **D:**
Quality Attributes	[10 Marks] **A:** **B:** **C:** **D:**
Plan for Testing Levels	[10 Marks] **A:** **B:** **C:** **D:**
Test cases, Format, Submission	[20 Marks] **A:** **B:** **C:** **D:**

Software Test Plan

for

<Project>

Version 1.0 approved

Prepared by <author>

<organization>

<date created>

Table of Contents

1. TEST PLAN IDENTIFIER: AT-TP01.3

2. REFERENCE MATERIALS

The following documents are referenced for the development and testing of **AMR Predictor Pro**:

1. **Software Requirement Specification (SRS) Document**: This document outlines the functional and non-functional requirements for the AMR Predictor Pro tool, detailing system features such as data input, prediction modules, machine learning integration, and user management.
2. **User Manual**: A guide for end-users providing instructions on how to operate AMR Predictor Pro, including data input, model training, prediction generation, and report exports.
3. **System Architecture Document**: A technical document detailing the architecture of the AMR Predictor Pro system, including the integration of machine learning modules, external database connections, and real-time data streaming capabilities.
4. **Test Case Document**: This document contains all the specific test cases designed for validating the functionality of AMR Predictor Pro, ensuring that each module operates according to the requirements outlined in the SRS.
5. **Regulatory Compliance Documents**: These include guidelines and regulations such as GDPR (for data privacy) and HIPAA (for healthcare-related data), ensuring that the system complies with global standards for data protection and security.
6. **Existing Studies and Research Papers**:
    - *Antimicrobial Resistance: A Global Health Threat* (2024): Discusses the current state of AMR and emphasizes the need for tools like AMR Predictor Pro.
    - *Using Bacterial Pan-Genome-Based Feature Selection Approach to Improve MIC Prediction* (2023): Provides insights into machine learning methods for antimicrobial resistance prediction.
    - *PathoFact: A Pipeline for Predicting Virulence Factors and AMR Genes* (2021): Discusses the use of bioinformatics pipelines for AMR gene detection, relevant for the software's gene detection features.

3. INTRODUCTION

3.1 Background to the Problem

Antimicrobial resistance (AMR) represents one of the most significant threats to global health today. Over decades, excessive commercialization and widespread misuse of antibiotics have accelerated the development of resistant pathogens. Historically, substances like mercury, lead, and radium were once

widely used despite their harmful effects, and similar patterns can be seen in the overprescription and misuse of antibiotics.

The overuse of antibiotics, both in clinical settings and agriculture, has contributed to the rise of multidrug-resistant bacteria, viruses, and fungi. Antibiotics once celebrated as a miracle solution to bacterial infections, are now facing diminishing returns as microorganisms evolve, rendering these drugs increasingly ineffective. This global health threat is evident in hospitals, where infections once easily treatable now result in prolonged illness and higher mortality rates. Furthermore, the rising costs associated with treating drug-resistant infections place a significant burden on healthcare systems worldwide.

Microorganisms have long evolved to produce antibiotic substances as a means of survival, out-competing other organisms for resources. However, over time, exposure to low doses of antibiotics has enabled them to adapt, developing resistance. As early as 1928, when Alexander Fleming discovered penicillin, scientists like him warned of the potential consequences of overusing antibiotics. Today, these warnings have materialized. We now face a global crisis where previously treatable infections are once again becoming deadly.

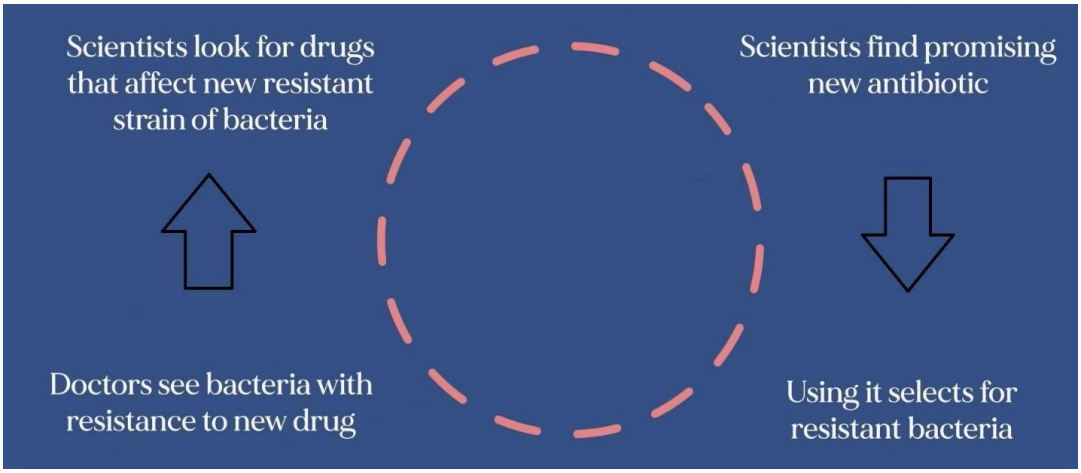


Figure [1]: The core problem of antibiotics [Right to Left].

3.2 Solution to the Problem

To address this pressing issue, there is an urgent need for innovative solutions that can predict and combat antimicrobial resistance. Traditional methods of discovering new antibiotics have slowed significantly, and alternative strategies are required. Current efforts include the identification of new antimicrobial-producing organisms, chemical modification of existing antibiotics, and exploring alternatives like bacteriophages and nanoparticles to enhance treatment efficacy.

In this context, “AMR Predictor PRO” emerges as a comprehensive scientific tool designed to assist researchers and healthcare professionals in predicting and managing antimicrobial resistance. Unlike traditional methods, AMR Predictor PRO integrates cutting-edge machine learning technology, enabling users to analyze large datasets, identify resistance patterns, and predict the efficacy of antibiotics in real-time.

The system provides several features essential to AMR research, including advanced data visualization tools, integration with external databases, and the ability to track bacterial mutations. One of its key strengths is the inclusion of AI-driven models, which can significantly reduce manual work and provide more accurate predictions based on real-time data streams. This allows for faster decision-making in clinical settings, ultimately improving patient outcomes and reducing the spread of resistant pathogens.

Existing studies, such as those on pan-genome-based approaches for predicting minimum inhibitory concentration (MIC) and the utilization of nanopore sequencing for detecting resistance genes, highlight the importance of integrating genomic data with machine learning models. AMR Predictor PRO builds on these foundations, offering researchers an all-encompassing platform for studying resistance mechanisms, making it an invaluable tool for combating AMR on a global scale.

In conclusion, AMR Predictor PRO is not just a software tool but a vital step in advancing the fight against antimicrobial resistance. By leveraging state-of-the-art technologies and integrating with existing research infrastructures, this platform provides an essential resource for global efforts to combat the AMR crisis. The tool promises to support the development of new treatments, aid in real-time surveillance, and contribute to global health strategies aimed at preserving the effectiveness of antibiotics for future generations.

4. REQUEIREMNT SPECIFICATION

4.1 System Features

1. Authentication and Authorization Module

Functional Requirement:

1.1 Users can log in using a valid username or Scientist ID and password, with inputs centered and clearly labeled.
1.2 The system shall support a "Forgot Password?" functionality, all to reset their credentials. The link should be underlined for easy recognition.
1.3 The system shall display a "Remember Me" checkbox for easier future access, with the label right-aligned next to the checkbox.
1.4 The system shall include a card scanner option, allowing users to insert a card for infrared scanning for secure login.
1.5 The system shall visually distinguish all input fields, buttons, and the scanner using borders and padding to enhance usability.

- **Priority Level:** High
- **Precondition:** User credentials or infrared card scanners must be valid.
- **Cross-Reference:** 9, 10.

2. Dashboard

Functional Requirement:

2.1 The system shall display a summary of the user's recent activities, including model training, predictions, and data analysis.
2.2 The system shall show user-specific quick stats, such as model accuracy, processed data, and the number of completed analyses, displayed using charts and graphs.
2.3 The system shall provide a collapsible navigation menu that allows users to access different sections of the software, including Data Input, Predictions, Model Training, Reports, AMR Gene Database, and other relevant sections.
2.4 The system shall allow users to view and edit their profile information, including their name, role, and last login details, from the user profile section in the dashboard.
2.5 The system shall include a search bar within the header to enable users to search for data or information across the application.
2.6 The system shall display unread notifications via a notification icon in the header, allowing users to view or dismiss notifications.
2.7 The system shall provide an overview section with charts and graphs summarizing the user's activities, including recent data input, model training, and SNP analysis.
2.8 The system shall provide access to additional sections like AMR Gene Database, SNP Analysis, Prediction Models, and Bacterial Growth (Biofilm Module) via the Navigation Menu.

- **Priority Level:** Medium
- **Precondition:** The user must be logged in.
- **Cross-Reference:** 3, 5.

3. Data Input Section

Functional Requirement:

3.1 Users can upload genetic data files.
3.2 Users can input clinical data through a form.
3.3 The system shall provide options for data preprocessing (e.g., normalization).
3.4 The system shall validate the uploaded data for completeness and format.
3.5 The system shall provide a data preview feature to inspect uploaded data.

- **Priority Level:** High
- **Precondition:** Data files should follow the prescribed format.
- **Cross-Reference:** 4, 6.

4. Model Training Section

Functional Requirement:

4.1 Users can upload and select pre-uploaded datasets for model training.
4.2 The system shall allow users to view the uploaded dataset in a table format and remove unwanted columns before training.
4.3 The system shall allow users to choose a machine learning algorithm from a dropdown list (e.g., Random Forest, SVM), with a brief description of each algorithm.
4.4 Users can configure hyperparameters through a form for the selected algorithm before training the model.
4.5 The system shall provide real-time progress bars and status updates during the training process.
4.6 The system shall allow users to save trained models, and the saved models will be listed in a table format with options to download.
4.7 Users can evaluate the trained model using metrics such as accuracy, precision, recall, and F1 score, and these metrics shall be displayed in a table format.
4.8 Users can compare multiple trained models using the "Model Comparison" feature, with performance metrics (Accuracy, Precision, Recall, F1 Score) displayed side by side.
4.9 The system shall maintain logs of all actions performed during model training, evaluation, and comparison.

- **Priority Level:** High
- **Precondition:** Preprocessed data must be available for training.
- **Cross-Reference:** 3, 5, 12.

5. Prediction Interface

Functional Requirement:

5.1 Users can input new genetic data through the Genetic Data Form, which includes fields for Gene Name, Gene Sequence, Mutation Type, Mutation Frequency, and Antibiotic Resistance.
5.2 Users can input clinical data through the Clinical Data Form, which includes fields for Patient ID, Age, Gender, Clinical Symptoms, Date of Diagnosis, and Antibiotic Treatment.
5.3 The system shall generate a prediction based on the input data and display the following in the Instant Insights section: Predicted Antibiotic Resistance, Probability of Resistance, and Recommended Antibiotic Treatment.
5.4 The system shall display a confidence score for the prediction along with a confidence interval.
5.5 Users can choose to save and export the prediction results in JSON, CSV, or PDF format using Save and Export Options.
5.6 The system shall allow users to select drug classes from the Drug Class Menu (e.g., Beta-Lactams, Macrolides, Tetracyclines) to further refine the predicted antibiotic treatment.
5.7 The system shall display a suggested medicine class based on the prediction in the Suggested Medicine Class Menu (e.g., Penicillins, Cephalosporins, Carbapenems).

5.8 The system shall display a "Resistant to Drug" table with the genus and species name, epidemiological data, and corresponding antibiotic resistance data.

- **Priority Level:** High
- **Precondition:** The trained model must be available for prediction.
- **Cross-Reference:** 4, 6.

6. Reports Section

Functional Requirement:

6.1 The system shall allow users to generate new reports on model performance, predictions, and data analysis, with options to select specific data and customize the report format.
6.2 Users can view previously generated reports in the View Section, with filters to sort by date, type, and other relevant criteria.
6.3 The system shall provide options to export reports in multiple formats such as PDF, Excel, XML, and JSON, allowing users to select the format and customize the export content.
6.4 The system shall allow users to schedule automated reports in the Automated Scheduling Section, enabling regular report generation at specified intervals.
6.5 Users can customize report content and layout in the Customizability Section before generating, including data selection, format customization, and layout adjustments.
6.6 The system shall provide a Summary Section that includes charts and graphs to help visualize the data within the reports.
6.7 Users can share reports with other users and collaborate in the Sharing and Collaboration Section, with options to customize permissions for sharing and collaboration.
6.8 The system shall support integration with external databases, allowing users to include data from external sources in the reports.

- **Priority Level:** Medium
- **Precondition:** The system must have prediction or analysis data available to generate reports.
- **Cross-Reference:** 3, 5, 9.

7. User Management and Settings Interface

Functional Requirement:

7.1 Admin users can manage user roles and permissions (e.g., Admin, Researcher, Guest).
7.2 Users can change their password and enable multi-factor authentication (MFA).
7.3 Admin users can view user activity logs and audit trails.
7.4 Users can configure notification preferences.
7.5 Admin users can manage data storage and backup options.

- **Priority Level:** Medium
- **Precondition:** Admin users must have appropriate privileges.
- **Cross-Reference:** 1, 9.

8. Help and Support Interface

Functional Requirement:

8.1 Users can access help documentation such as user manuals, FAQs, and tutorials.
8.2 The system shall provide a support ticket system where users can submit and track issues.
8.3 The system shall offer real-time chat support with an agent.
8.4 Users can submit feedback and suggestions through a feedback form.

- **Priority Level:** Medium
- **Precondition:** The user must have access to the system.
- **Cross-Reference:** 1, 7.

9. Advanced Data Analysis and Visualization Interface

Functional Requirement:

9.1 The system shall provide data visualization tools such as heatmaps and ROC curves.
9.2 Users can filter and sort results based on time range, location, and pathogen type.
9.3 The system shall allow users to compare different models or datasets.
9.4 Users can export visualizations as images or interactive plots.
9.5 Users can save custom dashboard views for future use.

- **Priority Level:** Medium
- **Precondition:** The system must have data to visualize.
- **Cross-Reference:** 5, 6.

10. Security and Compliance Module

Functional Requirement:

10.1 The system shall encrypt all user data in storage and transit.
10.2 The system shall support two-factor authentication for enhanced login security.
10.3 Admin users can view and manage audit logs for compliance.
10.4 The system shall track user consent for data usage and manage consent accordingly.

- **Priority Level:** High
- **Precondition:** User data must be handled securely.
- **Cross-Reference:** 1, 7.

11. Integration with External Databases

Functional Requirement:

11.1 The system shall allow users to configure connections with external databases.
11.2 The system shall support data synchronization with external sources.
11.3 Users can query external databases for additional data.

- **Priority Level:** Medium
- **Precondition:** External database access credentials must be available.
- **Cross-Reference:** 3, 6.

12. Real-time Data Streaming

Functional Requirement:

12.1 Users can configure real-time data streams from internal sources, such as bacterial growth, bio-film analysis, gene sequencing activity, and external sources, such as third-party databases or sensors, using the system's configuration interface.
12.2 The system shall display live data in real-time using appropriate charts, including bacterial growth, bio-film analysis, gene sequencing, drug effectiveness, mutation tracking, and hazard monitoring.
12.3 The system shall allow users to adjust the time range displayed on the charts using a slider, providing control over the real-time data view.
12.4 The system shall allow users to select different strains, drugs, or sequences using dropdown menus, which will update the real-time chart display.
12.5 The system shall trigger real-time alerts and notifications based on user-defined thresholds for data conditions, including bacterial growth rate, drug effectiveness, mutation rate, and laboratory hazard levels.
12.6 The system shall provide options to save snapshots of the current data, export reports, and download raw data in real time, ensuring data capture and reporting flexibility.
12.7 Users can access containment logs and relevant data about laboratory hazards and containment status through the Alerts & Notifications section.

- **Priority Level:** Medium
- **Precondition:** Live data streams must be available.
- **Cross-Reference:** 4, 9.

13. User Activity Monitoring

Functional Requirement:

13.1 Admin users can view logs of all user activities.
13.2 The system shall monitor active sessions and provide details of each session.
13.3 The system shall generate real-time alerts for suspicious activities.

- **Priority Level:** High
- **Precondition:** Admin users must have appropriate access rights.
- **Cross-Reference:** 7, 10.

14. Automated Data Backup

Functional Requirement:

14.1 The system shall allow users to schedule regular automated data backups in various formats, such as CSV, Excel, and PDF, with options to specify the storage location (e.g., local drive, and cloud services like Dropbox or Google Drive).
14.2 Users can manually initiate data backups at any time, with the option to select the format and storage location.
14.3 The system shall display a progress bar during the backup process, indicating the current status and estimated completion time.
14.4 Users shall be able to restore previously backed-up data from the specified storage location.
14.5 The system shall provide options for configuring archival policies, allowing users to set retention periods for backups and manage old backup files.
14.6 The system shall allow users to receive notifications and alerts for successful backup completion or errors in the process.

- **Priority Level:** Medium
- **Precondition:** Adequate storage space must be available for backups.
- **Cross-Reference:** 10, 6.

15. Customizable Workflow Management

Functional Requirement:

15.1 Users can create new workflows by combining different modules.
15.2 The system shall provide workflow templates for common processes.
15.3 Users can configure automation rules for the workflows.

- **Priority Level:** Medium
- **Precondition:** Predefined workflow templates should be available.
- **Cross-Reference:** 3, 4.

16. Multi-language Support

Functional Requirement:

16.1 The system shall provide a dropdown for users to select their preferred language.
16.2 Admin users can manage translations for different languages.
16.3 The system shall allow users to configure localization options like date and time format.

- **Priority Level:** Low
- **Precondition:** Translation resources for the supported languages must be available.
- **Cross-Reference:** 7, 8.

17. Notification Center

Functional Requirement:

17.1 The system shall display a list of all notifications for the user.
17.2 Users can mark notifications as read or unread.
17.3 Users can configure their notification preferences (e.g., email, SMS, in-app notifications).

- **Priority Level:** Medium
- **Precondition:** Users must have permission to receive notifications.
- **Cross-Reference:** 12, 9.

18. API Access and Integration

Functional Requirement:

18.1 The system shall provide API documentation for external integration.
18.2 Users can generate and manage API keys for secure access to APIs.
18.3 The system shall enforce rate limits for API requests.

- **Priority Level:** High
- **Precondition:** API access credentials must be configured.
- **Cross-Reference:** 11, 7.

4.2 System Quality Attributes

QA1 - Usability:

- A researcher shall be able to input genetic and clinical datasets and generate AMR predictions within an average of three minutes, with intuitive navigation and minimal training.

Priority Level: High
Precondition: User must have a valid account and data to input.
Cross-Reference: 1, 2

QA2 - Performance:

- The system should handle large-scale genomic datasets and provide AMR prediction results within 5 seconds under normal load (up to 100,000 genes in the dataset).
- Data visualizations and report generation must occur within 3 seconds for real-time scientific analysis.

Priority Level: High
Precondition: The system must have sufficient server capacity.
Cross-Reference: 4, 5

QA3 - Scalability:

- The platform must scale to support large research institutions with up to 5,000 concurrent users and handle petabyte-scale genomic data without significant performance degradation.

Priority Level: Medium
Precondition: The system should be deployed with adequate cloud or server infrastructure.
Cross-Reference: 9, 14

QA4 - Reliability:

- The system shall maintain a minimum of 99.99% uptime to ensure continuous availability for global scientific research and collaborations.

Priority Level: High
Precondition: Redundant server infrastructure should be in place.
Cross-Reference: 10, 16

QA5 - Security:

- Sensitive genomic and clinical data must be encrypted at rest and in transit using AES-256 encryption.
- Access controls must ensure that only authorized scientists or clinicians can access or modify sensitive data.
- Two-factor authentication (2FA) shall be mandatory for all users accessing the prediction module.

Priority Level: High
Precondition: User accounts should be provisioned with proper access control.
Cross-Reference: 1, 10

QA6 - Data Integrity and Accuracy:

- The system shall ensure data integrity by verifying the correctness of input files and datasets, preventing incomplete or erroneous data from affecting the AMR predictions.
- Prediction accuracy must be validated against external benchmark datasets with a minimum accuracy threshold of 95%.

Priority Level: High
Precondition: Valid datasets and external benchmarks must be available.
Cross-Reference: 3, 12

QA7 - Maintainability:

- The system shall be designed for continuous updates and upgrades without downtime, ensuring that scientific and regulatory compliance is maintained as research evolves.
- Any system issues or bugs reported by users should be resolved and deployed within 24 hours to ensure minimal disruption to research activities.

Priority Level: Medium
Precondition: Active development and support team should be available.
Cross-Reference: 1, 5

QA8 - Serviceability:

- Scientists and researchers should be able to request support for data-related issues, with guaranteed response times within 2 hours for critical cases and 12 hours for standard requests.
- A help desk system shall be available for live troubleshooting and resolution of technical issues.

Priority Level: Medium
Precondition: Help desk and support system must be active.
Cross-Reference: 8, 16

QA9 - Interoperability:

- The system must integrate with international genomic databases (such as GenBank and ENA) and comply with scientific data standards like FASTA, VCF, and GFF formats.

- The software must support data exchange and collaboration through secure APIs with research platforms like Galaxy, BLAST, and Nextflow.

  Priority Level: Medium
  Precondition: Proper APIs and connectors must be configured.
  Cross-Reference: 9, 18

QA10 - Data Backup and Recovery:

- The system shall perform daily automated backups of all genomic and clinical data.
- In the event of a system failure, full data recovery must occur within 4 hours to prevent loss of scientific research.

  Priority Level: High
  Precondition: Sufficient storage for backup and recovery must be available.
  Cross-Reference: 14

QA11 - Regulatory Compliance:

- The platform shall comply with scientific data protection regulations, including GDPR for European users and HIPAA for U.S.-based healthcare data, ensuring ethical handling of sensitive genomic and clinical data.
- The system should also support compliance with FAIR (Findable, Accessible, Interoperable, and Reusable) data principles.

  Priority Level: High
  Precondition: Compliance checks and audits must be part of the deployment strategy.
  Cross-Reference: 10, 15

QA12 - Scientific Integrity:

- The system shall log all user activities, such as data uploads, model training, and predictions, to ensure full traceability for scientific reproducibility and auditing.
- All modifications to data or predictions must be documented and attributed to the user who made the changes.

  Priority Level: High
  Precondition: A logging and auditing system must be implemented.
  Cross-Reference: 1, 12

QA13 - Visualization and Analytical Reporting:

- Data visualizations (e.g., heatmaps, genomic data trends, resistance patterns) must be generated in under 2 seconds to support real-time scientific decision-making.
- Analytical reports must include customizable data filters and be exportable in standard scientific formats (PDF, CSV, and Excel).

Priority Level: Medium
Precondition: The system must have prediction or analysis data available for visualization.
Cross-Reference: 5, 6

QA14 - Scalability for Collaboration:

- The system must support multiple research teams across different institutions with real-time data sharing and joint analysis functionalities, allowing for up to 10 teams working on the same dataset simultaneously without any conflict or delay in updates.

Priority Level: Medium
Precondition: Cloud-based collaboration infrastructure must be in place.
Cross-Reference: 9, 14

QA15 - Data Anonymization:

- Clinical data involving patient information must be anonymized by default before sharing with external research collaborators to comply with ethical standards and data protection laws.

Priority Level: High
Precondition: Anonymization and de-identification algorithms must be configured.
Cross-Reference: 5, 11

QA16 - Availability:

- The system must ensure 24/7 availability to support global research efforts, with downtime scheduled for less than 1 hour per month for maintenance activities.

Priority Level: High
Precondition: Redundant infrastructure must be available.
Cross-Reference: 4, 10

4.3 Project Requirements

COCOMO I, the original Constructive Cost Model developed by Barry Boehm in 1981, is not widely used anymore due to several significant limitations. The first calibration of COCOMO II was released under the name "COCOMO II.1997" and aimed to address the limitations of the original model by incorporating more modern software development practices and a wider range of project variables.

Reasons for not using the Basic COCOMO Model:
1. It considers only KLOC to estimate effort.
2. It cannot be used for modern tools.
3. It does not consider different parameters.

Budget Complications:

Budgeting for AMR (Antimicrobial Resistance) tools is a complex process. Such funding typically falls under laboratory equipment allocations within broader budgets aimed at developing new antibiotic solutions. The overall financing of antibiotic research is intricate, often involving a mix of public and private sources. Government-backed initiatives, international research institutes, and collaborations between academia and industry frequently contribute to these funds. Additionally, patent agreements and royalties can introduce further financial complexities. In some cases, crowdfunding efforts are employed to support specific research projects. As scientific tools and techniques continue to evolve, laboratories must adapt their budgets to accommodate advancements in AMR research technologies and methodologies.

To estimate the time required to develop an **AMR (Antimicrobial Resistance) Tool** using a model like the **Advanced COCOMO** (Constructive Cost Model), we need to break down the process into several key phases. The COCOMO model estimates effort and time based on the complexity, size, and the required precision of the software, and it uses different levels of granularity (Basic, Intermediate, or Detailed).

For a **scientific precision tool** like the AMR Predictor PRO, we would likely be using **COCOMO II**, which is an advanced version that accounts for modern software development practices and systems.

Required Inputs for COCOMO II:

1. **Size of the Project**:
    - Measured in **Kilo Lines of Code (KLOC)**. Scientific tools, especially with machine learning modules, data integration, and visualization, are often quite large. Let's estimate:
        - For an advanced AMR tool with machine learning, data analysis, prediction models, and a detailed UI: ~**100 KLOC**.
2. **Scale Factors**:

- **Precedentedness (PREC)**: Developing scientific tools like this may have some historical precedence but will also require innovative approaches, so this could be moderate.
- **Development Flexibility (FLEX)**: Since this tool needs high precision, flexibility will be low.
- **Team Cohesion (TEAM)**: Assumed to be moderately good for scientific development teams.
- **Process Maturity (PMAT)**: For a high-precision tool in a biohazardous lab, maturity and quality control will be essential, so high PMAT.

3. **Cost Drivers (Effort Multipliers)**:
   - **Product Complexity (RCPX)**: The complexity of developing the machine learning modules and integrating them into real-time data analysis is high.
   - **Required Reliability (RELY)**: Very high reliability is expected since the tool will be used in critical scientific environments.
   - **Personnel Capabilities (PERS)**: Skilled scientists and developers with expertise in bioinformatics and software engineering will likely be part of the team, so this will be high.

Applying Advanced COCOMO II Formula:

1. **Effort (Person-Months)**: The formula for COCOMO II is:

$$\textbf{Effort} = A \times (\text{Size})^{B} \times \prod EM_i$$

Where:

- **A** is a constant (usually around 2.94 for modern development projects).
- **B** is an exponent derived from the scale factors.
- **Size** is the size of the project in KLOC (estimated at 100 KLOC).
- EM_i are the effort multipliers (cost drivers based on project, team, and technical factors).

2. **Time to Develop (TDEV)**: After calculating effort in person-months, the time to develop can be estimated using:

$$\text{TDEV (months)} = 3.67 \times (\text{Effort})^{0.28}$$

Example Estimate (Simplified):

Assuming **100 KLOC** and medium-high complexity for an advanced AMR tool:

1. **Effort**: If we assume an effort of around **600 person-months** based on the size and complexity of the project (estimated from a similar scientific tool with data integration, machine learning models, and high reliability), the equation would yield:
2. **Time to Develop**: Using the effort calculation:

   $$\mathbf{TDEV = 3.67 \times (600)^{0.28} \approx 28}$$

Refinement Based on Advanced Features:

- Since this is a **precision scientific tool** with real-time data streaming, prediction models, and integration with external databases, it's possible that additional months would be required for testing, regulatory compliance, and data validation.
- Factoring in these elements could lead to a development timeline of approximately **2.5 to 3 years** for completion, assuming a team of **10-15 developers and researchers** working continuously.

Summary:

- **Effort**: ~**600 person-months** (based on complexity and KLOC).
- **Time to Develop**: ~**28-36 months** (depending on external dependencies and testing phases).

This estimate provides a baseline using the COCOMO II model for a high-complexity, precision-oriented scientific tool like the AMR Predictor PRO.

The development of a **scientific precision tool** like the **AMR Predictor PRO** would fall under the **Post-Architectural** phase of the **COCOMO II** model.

Explanation:

COCOMO II consists of three submodels, each applied at different stages of a project:

1. **Application Composition Model**: Used during early prototyping or when building systems with significant reuse of components. It is less suitable for highly specialized scientific tools like AMR systems.
2. **Early Design Model**: Used after basic requirements are gathered and high-level architectural decisions are made. This model is used to estimate at a stage when detailed requirements are not yet fully developed but major subsystems are identified.
3. **Post-Architectural Model**: Applied once the system's architecture is established, and detailed requirements and design components are fully defined. It is more precise and accounts for cost drivers such as reliability, complexity, and the use of real-time data streams, machine learning modules, and integration with external databases—features critical for the AMR Predictor PRO.

Why Post-Architectural?

- **Detailed Requirements**: For a complex scientific tool, most of the key components (e.g., machine learning models, real-time data streaming, prediction modules) are likely already identified and defined in detail. The project is at a point where these specific modules need to be built, integrated, and fine-tuned.
- **Architecture Defined**: By this stage, the system architecture should be largely defined, covering data flows, user interfaces, and processing pipelines necessary for scientific precision.
- **Cost Drivers**: The Post-Architectural model uses a comprehensive set of **cost drivers** (like product complexity, reliability, performance constraints, and personnel capability), which are important for such a high-precision tool. These drivers are not used in early design, where the architecture is not fully established.

Therefore, the **Post-Architectural model** in **COCOMO II** is the most appropriate for estimating the effort and time required to develop the AMR Predictor PRO, especially given its complexity and precision requirements.

To calculate a **precise budget** for the development of an **AMR (Antimicrobial Resistance) Tool** like **AMR Predictor PRO** using the **COCOMO II Post-Architectural Model**, we can break down the cost estimation into several key components. The primary factors influencing the budget include:

1. **Effort (in person-months)**: The total effort required to complete the project.
2. **Salary/Cost of Personnel**: The average cost of developers, scientists, and other personnel working on the project.
3. **Overhead Costs**: Including software licenses, hardware, laboratory costs, cloud services, etc.
4. **Testing and Compliance**: Scientific tools like AMR systems require rigorous testing, validation, and sometimes regulatory compliance, which adds to the cost.
5. **Additional Expenses**: Other expenses include research material, datasets, data storage, and external tools.

Step-by-Step Example to calculate the budget using **COCOMO MODEL II:**

1. Effort Calculation Using COCOMO II (Post-Architectural Model)

As calculated in the previous steps, an estimated **600 person-months** of effort are required based on a **100 KLOC (Kilo Lines of Code)** project with medium-to-high complexity.

2. Average Cost per Personnel

In a high-precision scientific project, personnel costs are substantial. Here's a breakdown:

- **Developers and Engineers**: $10,000/month

- **Data Scientists and Bioinformaticians**: $12,000/month
- **Project Managers/Lead Scientists**: $15,000/month

Total Average Personnel Cost:

Assuming a team composition of:

- **8 Developers/Engineers**: 8 × 10,000 = 80,000/month
- **3 Data Scientists**: 3 × 12,000 = 36,000/month
- **1 Project Manager/Lead Scientist**: 1 × 15,000 = 15,000/month

Total Personnel Cost/Month:

80,000 + 36,000 + 15,000 = 131,000/month

3. Total Personnel Cost for the Project

Now, using the effort calculated earlier (600 person-months), assuming this team works **12 person-months per month** (i.e., 12 people):

Total Time = 600 person-months / 12 people = 50 months

Now, calculate total personnel costs for the duration:

Personnel Cost = 131,000 × 50 = **6,550,000**

4. Overhead Costs

- **Hardware/Infrastructure**: $500,000 (high-performance servers, GPUs for machine learning, etc.)
- **Software/Tool Licenses**: $300,000 (bioinformatics tools, machine learning platforms, etc.)
- **Cloud Services for Data Storage and Processing**: $250,000 (storage of genome sequencing, model training in the cloud)
- **Testing and Validation Tools**: $200,000

Total Overhead Costs:

500,000 + 300,000 + 250,000 + 200,000 = **1,250,000**

5. Testing, Compliance, and Validation

Given the biohazardous nature of the data and the precision required for scientific tools, additional costs arise from:

- **Data Validation and Testing**: $500,000
- **Regulatory Compliance**: $200,000 (adhering to lab safety and biohazard protocols)

- **Scientific Peer Review and Verification**: $150,000

Total Testing and Compliance Costs:

500,000 + 200,000 + 150,000 = **850,000**

6. Additional Research and Datasets

Research datasets (e.g., pathogen sequences, resistance data), scientific papers, and external collaborations may add further expenses:

- **Research Material and Datasets**: $300,000
- **Collaborations with External Institutes**: $200,000

Total Additional Costs:

300,000 + 200,000 = **500,000**

7. Total Estimated Budget

Now, we add up all the costs to get the total budget estimate:

- **Personnel Cost**: $6,550,000
- **Overhead Costs**: $1,250,000
- **Testing and Compliance**: $850,000
- **Research and Datasets**: $500,000

Total Project Budget:

6,550,000 + 1,250,000 + 850,000 + 500,000 = **9,150,000**

Final Estimate:

- **Total Budget**: ~**$9.15 million** over 50 months

Factors Affecting the Budget:

- **Team Size**: Increasing the team size could reduce development time but increase the monthly budget.
- **Complexity**: If the AMR tool includes additional features, such as integrating with more external databases, advanced visualization, or higher precision in prediction models, this may increase the effort and costs.
- **Location**: Development costs might vary based on geographic location and the cost of personnel.

5. TESTING APPROACH

Testing Levels

This report offers a comprehensive overview of the test levels for AMR Predictor PRO, a sophisticated software tool designed to aid in the analysis and prediction of antimicrobial resistance. To ensure the long-term sustainability of this project, a structured approach has been adopted beginning with a foundational examination of the project's scope and objectives. More refinements will be added in the future.

Roadmap of All Considered Test Levels for AMR Predictor PRO:

1. Requirement Analysis and Test Planning Phase
2. Unit Testing
3. Integration Testing
4. System Testing
5. Regulatory and Compliance Testing
6. User Acceptance Testing (UAT)
7. Special Testing (Anomaly and AI Validation)
8. Post-Deployment Monitoring and Maintenance Testing
9. Change Request (CR) and Future Version Testing

All these falls under the special case of AMR Tool and the context depends along with the fundamentals of testing levels. This roadmap ensures that AMR Predictor PRO undergoes rigorous testing at every stage of development and adapts over time as science and technology evolve, while complying with long-term funding and regulatory constraints.

1. Requirement Analysis and Test Planning Phase

Initial Starting Point: SRS (Software Requirement Specification) Document

- **Goal**: Define requirements and testing needs based on the initial version of the SRS document.
- **Key Activities**:
    - Analyze SRS and establish test requirements.
    - Define success metrics, KPIs, and test scenarios.

- Determine the appropriate testing strategy (unit, integration, system, user acceptance, and regulatory testing).
- Identify necessary tools for automation and data validation.
- Consider long-term support needs in the testing plan.

- **Deliverable:** Comprehensive Test Plan Document.

Testing of the SRS Doc can be automated as well as manual. For this special case we can select hybrid approach.

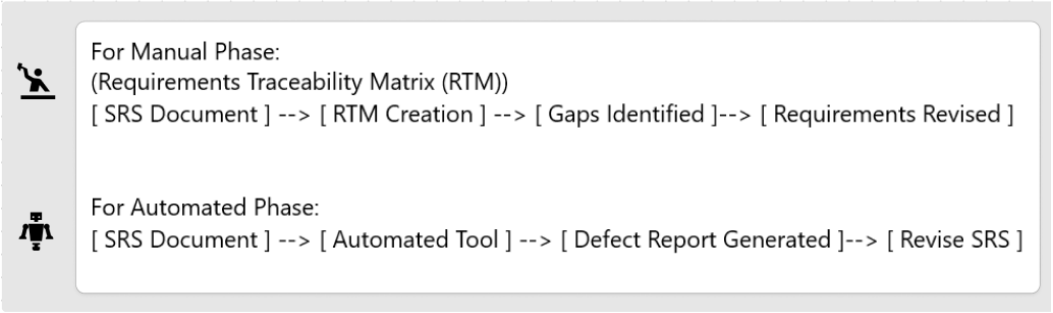


Figure [2]: Considered Testing types for SRS Dicument of AMR Predictor PRO.

2. Unit Testing

- **Goal**: Validate the smallest individual components and algorithms used for AMR predictions.
- **Key Activities**:
  - Develop unit test cases for every microservice or algorithm module.
  - Automate unit testing for frequently evolving components.
  - Ensure biological data (e.g., gene sequences, pathogen profiles) is processed accurately.
- Deliverable: Verified and validated code modules.

Detailed Planning of Unit testing for 18 core features:

1. Authentication and Authorization Module

Component: Username Field

- **Goal:** Ensure the username field validates inputs correctly.
- **Considerations:**

- **Empty Input:** Testing how the system behaves when the username field is blank. The system should return an error indicating that the field cannot be empty.
- **Invalid Format:** Testing usernames that do not follow the correct email format (user@, user@123). The system should flag these as invalid.
- **Valid Format:** We can test correctly formatted usernames (user@example.com) to confirm that they pass validation.

Component: Password Field

- **Goal:** Ensure password validation and security.
- **Considerations:**
  - **Empty Input:** Ensuring an empty password triggers an error message.
  - **Weak Passwords:** Testing for password complexity (short passwords, passwords without special characters).
  - **Valid Password:** Testing complex passwords to confirm they pass validation.

Component: Login Button

- **Goal:** Ensure login functionality works properly.
- **Considerations**:
  - **Valid Login:** Testing the login flow with a valid username and password to ensure the system grants access and redirects to the dashboard.
  - **Invalid Login:** Testing with incorrect credentials and ensuring the system returns the appropriate error message.

Component: Remember Me Checkbox

- **Goal:** Ensuring the "Remember Me" checkbox correctly handles user sessions.
- **Considerations:**
  - **Default State:** The checkbox should be unchecked by default when the login screen is loaded.
  - **Check and Login State:** Verify that the session remains active without requiring re-authentication on subsequent visits once the user comes in with the "Remember Me" option selected.
  - **Unchecked State:** If the user logs in without checking the "Remember Me" box, the session should expire when the browser is closed or after a set inactivity period.
  -

2. Dashboard

Component: Navigation Menu

- **Goal:** Ensure correct navigation between sections.
- **Considerations:**
    - **Correct Section Loading:** When a menu item is clicked confirm that the correct section loads.
    - **Active State:** Ensuring that the clicked menu item is visually indicated as active.

Component: Quick Stats Widget

- **Goal:** Ensures correct display of key statistics (i.e., Laboratory Pathogen Stats, Temperature at Lab).
- **Considerations:**
    - **Accurate Data Display:** After performing actions such as model training or data input, confirm that the widget displays accurate, updated statistics.
    - **Edge Cases:** Handle cases where no data or activities exist yet, ensuring the widget doesn't crash or display incorrect info.

3. Data Input Section

Component: Genetic Data Upload

- **Goal:** Ensure correct functionality of the file upload.
- **Considerations:**
    - **Valid File Upload:** Test the upload of valid file types (csv, xlsx, xml, json, etc.) and ensure the file uploads successfully.
    - **Invalid File Upload:** Upload unsupported file types and ensure the system correctly returns an error.
    - **Empty File:** Ensuring an empty file trigger a proper warning or error.

Component: Preprocessing Options

- **Goal:** Ensure that preprocessing functions are working.
- **Considerations:**
    - **Normalization and Scaling:** Making sure when a specific preprocessing option is selected, the system correctly applies the transformation to the data.
    - **Error Handling:** Ensuring the system handles preprocessing errors, such as missing data or mismatched data formats.

4. Model Training Section

Component: Select Training Data

- **Goal:** Ensure that selecting training data behaves as expected. Ensuring data quality and warning if faults in datasets are discovered.
- **Considerations:**
    - **Valid Dataset Selection:** When a valid dataset is selected, ensure it is properly loaded for training.
    - **Invalid Dataset:** Testing with invalid or empty datasets and ensuring the system returns an error or warning.

Component: Algorithm Selection

4. **Goal:** Ensure that users can select an algorithm and train models correctly.
5. **Considerations:**
    - **Valid Algorithm Selection:** Selecting various algorithms and ensuring they are applied during model training.
    - **Incompatible Algorithms:** Handling cases where an algorithm may not be compatible with the dataset and ensuring proper error messages are returned.

Component: Training Progress Bar

- **Goal:** Ensure the training progress bar functions correctly.
- **Considerations:**
    - **Progress Updates:** Confirm that the progress bar updates correctly as the model trains.
    - **Edge Cases:** Ensuring the bar reaches 100% when training completes and doesn't freeze or misreport progress.

5. Prediction Interface

Component: Input Data

- **Goal:** Ensure correct input of genetic and clinical data.
- **Considerations:**
    - **Valid Data:** Enter valid genetic and clinical data, and ensure the system accepts it.
    - **Invalid Data:** Testing the input of incorrect or incomplete data, ensuring the system returns relevant error messages.

Component: Generate Prediction

- **Goal:** Ensure predictions are generated correctly. Ensuring prediction matches laboratory results and can be replicated by other firms for better screening.
- **Considerations:**
    - **Correct Prediction Output:** After valid data input, ensure the correct prediction is displayed (resistance to a specific drug).
    - **Invalid Prediction Request:** Test with incomplete or invalid data and ensure the system doesn't generate predictions and instead returns an error.

Component: Confidence Interval

- **Goal:** Ensure the prediction confidence is correctly calculated.
- **Considerations:**
    - **Correct Confidence Calculation:** Verify that the confidence interval shown corresponds to the system's internal calculations once a prediction has been generated.

6. Reports Section

Component: View Reports

- **Goal:** Ensure report viewing and generation work correctly.
- **Considerations:**
    - **Valid Reports:** Make sure that after performing tasks like model training or prediction, reports are generated accurately.
    - **No Reports Available:** When there are no reports available, manage edge scenarios to ensure that the user receives correct and error-free prompts from the system.

Component: Export Options

- **Goal:** Ensure reports can be exported in various formats.
- **Considerations:**
    - **Successful Export:** Test exporting reports in formats such as [PDF, docx, Excel, etc.] ensuring the reports download correctly.
    - **Failed Export:** Simulate a failed export and ensure proper error handling.

7. User Management and Settings Interface

Component: User Roles and Permissions

- **Goal:** Ensure role-based access of users is correctly enforced.
- **Considerations:**

- **Admin Permissions:** Testing admin accounts and making sure they have access to all functionalities.
- **Restricted Roles:** Testing roles such as researcher or Guest and ensuring they only have access to specific sections.

Component: Password Management

- **Goal:** Ensure password updates work as expected.
- **Considerations:**
  - **Valid Password Update:** Ensure a valid password update succeeds, and the user is notified.
  - **Invalid Password:** Handling cases where users attempt to set weak passwords, ensuring the system shows message to choose stronger ones.

8. Help and Support Interface

Component: Support Ticket System

- **Goal:** Ensure ticket submission and tracking works.
- **Considerations:**
  - **Submit Ticket:** Test submitting valid support tickets and ensure they are correctly stored.
  - **View Tickets:** Ensuring users can view previously submitted tickets and their statuses.

9. Advanced Data Analysis and Visualization

Component: Visualization Tools

- **Goal:** Ensure various visualizations like heatmaps and ROC curves work correctly.
- **Considerations:**
  - **Generate Heatmap:** Generating a heatmap from valid data are going to be tested, to ensure it renders correctly.
  - **ROC Curve:** Test ROC curve generation, confirming accurate representation based on model performance.

10. Security and Compliance

Component: Data Encryption

- **Goal:** Ensure data encryption functionality works.
- **Considerations:**
  - **Encrypt Data:** Sensitive data is encrypted before storage will be tested.

- **Decrypt Data:** Ensuring data is properly decrypted when accessed by authorized users.

11. Integration with External Databases

- **Goal**: Ensure seamless integration with external scientific databases.
- **Considerations:**
  - **Database Connection Configuration:** The "Configure Connections" interface for selecting and saving connection settings will be validated.
  - **Data Synchronization:** Data synchronization is triggered correctly when the "Sync Data" button is clicked will be ensured.
  - **Query Interface:** Ensuring that correct error messages are displayed for invalid searches and that valid queries get the expected results.

12. Real-time Data Streaming

- **Goal**: Enable real-time data streaming and analysis.
- **Considerations:**
  - **Stream Setup:** The functionality of the "Setup Stream" button will be verified to ensure correct input validation.
  - **Live Data Visualization:** The "View Stream" feature updates the visualization in real time as new data is ingested will be ensured.
  - **Alerts and Notifications:** Making sure that while real-time data is being ingested, notifications are set off according to predetermined criteria.

13. User Activity Monitoring

- **Goal**: Track user activities for auditing and security.
- **Considerations:**
  - **View Logs:** Make sure all user actions such as logins and data uploads are accurately recorded in the activity logs.
  - **Session Monitoring:** Verify the list of active sessions, ensuring each user session is tracked properly.
  - **Real-time Alerts:** Ensure that alerts are triggered for suspicious activities (failed logins, data modification attempts).

14. Automated Data Backup

- **Goal**: Ensure data is backed up regularly and automatically.
- **Considerations:**
  - **Backup Scheduling:** Validate that the "Set Schedule" feature allows for scheduling backups at user-defined intervals.

- **Manual Backup:** Verify that clicking the "Backup Now" button triggers an immediate backup.
- **Restore Data:** Validate that the "Restore" feature allows the system to restore data from previous backups.

15. Customizable Workflow Management

- **Goal**: Create and manage customized workflows.
- **Considerations:**
  - **Workflow Creation:** Ensure that the "Create New Workflow" feature allows users to define workflows with valid steps and configurations.
  - **Workflow Templates:** Check that the pre-made workflow templates can be properly loaded and viewed.
  - **Automation Rules:** Validate that automation rules can be created and applied to workflows.

16. Multi-language Support

- **Goal**: Support multiple languages to enhance accessibility.
- **Considerations:**
  - **Language Selector:** Ensure that the language dropdown correctly lists all supported languages and changes the interface language accordingly.
  - **Translation Management:** Verify that custom translations can be added or edited for each supported language.
  - **Localization Options:** Validate that localization options change according to the selected language.

17. Notification Center

- **Goal**: Centralized notification system for alerts and updates.
- **Considerations:**
  - **Notification List:** Make sure that the read and unread states of each notification are clearly shown.
  - **Read/Unread status:** Make sure that all notification statuses are updated after selecting the "Mark All as Read" button.
  - **Notification Preferences:** Validate that users can configure their notification preferences (email, in-app).

18. API Access and Integration

- **Goal**: Provide APIs for third-party integration.
- **Considerations:**

- **API Documentation:** Ensure that the API documentation is accessible and displays the correct usage details for each API endpoint.
- **API Key Management:** Check if users can create, view, and cancel API keys.
- **API Rate Limits:** Simulate multiple API requests and ensure that rate limits are enforced properly.

Here, only the micro parts of code like (working principle of functions within a class) for all 18 features will be considered. (Precisely at level 1, lowest level)

Example Code of Mock Unit Test in python.

```python
import unittest

class Medicine:
    def __init__(self, infection_type):
        self.infection_type = infection_type

    def sol_mrsa_treatment(self):
        if self.infection_type == "MRSA":
            return "Glycopeptides: Vancomycin and teicoplanin are often considered first-line options.\n" \
                   "Cephalosporins: Ceftobiprole and ceftaroline are newer cephalosporins with activity against MRSA.\n" \
                   "Daptomycin: A lipopeptide antibiotic with bactericidal activity against MRSA.\n" \
                   "Linezolid: An oxazolidinone antibiotic with activity against MRSA and other Gram-positive bacteria."
        else:
            return "No specific recommendation available."

# Unit testing class
class TestMedicine(unittest.TestCase):
    def test_mrsa_treatment(self):
        # Create a Medicine object for MRSA infection
        mrsa_case = Medicine("MRSA")

        # Assert the expected output
        self.assertEqual(mrsa_case.sol_mrsa_treatment(), "Glycopeptides: Vancomycin and teicoplanin are often considered first-line options.\n"
                                                         "Cephalosporins: Ceftobiprole and ceftaroline are newer cephalosporins with activity against MRSA.\n"
                                                         "Daptomycin: A lipopeptide antibiotic with bactericidal activity against MRSA.\n"
                                                         "Linezolid: An oxazolidinone antibiotic with activity against MRSA and other Gram-positive bacteria.")

    def test_non_mrsa_treatment(self):
        # Create a Medicine object for non-MRSA infection
        non_mrsa_case = Medicine("non-MRSA")

        # Assert the expected output
        self.assertEqual(non_mrsa_case.sol_mrsa_treatment(), "No specific recommendation available.")

if __name__ == '__main__':
    unittest.main()
```

Output:

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.008s
OK
```

Figure [3]: Example based Unit Test by small function of AMR Predictor PRO in python.

3. Integration Testing

- **Goal**: Ensure that individual components work together as a whole.
- **Key Activities**:
  - Perform tests on data pipelines, prediction models, and interaction with databases (containing genomic or microbial data).

  - Test integrations with data sources (e.g., global AMR databases) and ensure data integrity.
  - Check communication between backend services and the user interface.
  - Include integration with external software like lab devices or cloud-based services.

- **Deliverable**: Functionally tested interconnected systems.

Detailed Planning for Integration under phases:

Identified Integration Points from core features

- Authentication Module ↔ Dashboard
- Dashboard ↔ Data Input Section
- Data Input Section ↔ Model Training Section
- Model Training Section ↔ Prediction Interface
- Prediction Interface ↔ Reports Section
- User Management ↔ Security Module
- API Access ↔ External Database Integration

Integration Testing Phases

Phase 1: Prepare Test Environment

- Create a test environment that is identical to the real one.
- Verify that all setups and dependencies are correct.

Phase 2: Define Integration Test Cases

- **Authentication and Authorization**: Check permissions, manage roles, and test login.
- **Dashboard**: Ensure that real-time data from the Data Input, Model Training, and Reports modules is shown in the overview area.
- **Data Input Section**: Verify data validation and file upload and confirm connection with the model training module.
- **Model Training Section**: Verify that training data integrates correctly with input from the Data Input section.
- **Prediction Interface**: Test data flow from Data Input to Prediction and ensure results display correctly.

- **Reports Section**: Check that reports pull data from Model Training and Prediction.
- **User Management**: Verify that user permissions affect accessibility across modules.
- **Help and Support**: Ensure support features are accessible throughout the application.

Phase 3: Execute Integration Testing approaches

1. **Top-Down Testing**

We can start from the Dashboard and user experience, ensuring higher-level components work before testing the data flows into lower-level modules.

2. **Bottom-Up Testing**

Begin with lower-level data operations such as Data Input, Model Training and gradually integrate them with high-level interfaces like the Dashboard.

3. **Incremental Integration**

We can gradually build the system by integrating modules based on their critical interactions by starting with the most important data workflows. Afterwards we can prioritize critical data flow such as Data Input → Model Training → Prediction Interface that can ensure smooth operation at each step before moving to the next module.

4. Big Bang Testing

As this approach is not recommended for large systems, we can skip this approach.

5. Sandwich Testing

This test can be done simultaneously for high-level user modules and low-level backend systems to ensure seamless interaction across the system.

Test Documentation

- Maintain comprehensive documentation of test cases, test results, and defects.
- Document the integration process and any lessons learned for future testing cycles.

Example Code of Mock Integration Test in python.

```python
import unittest
from unittest.mock import MagicMock

# Mock Database Class
class MockDatabase:
    def __init__(self):
        self.data = {
            'MRSA': {'resistance_rate': 90, 'antibiotics': ['Vancomycin', 'Linezolid']},
            'VRE': {'resistance_rate': 70, 'antibiotics': ['Daptomycin', 'Teicoplanin']}
        }

    def get_amr_data(self, infection_type):
        # Simulate retrieving data from a database
        return self.data.get(infection_type, None)

# AMR Tool Class
class AMRTool:
    def __init__(self, database):
        self.database = database

    def retrieve_and_process_data(self, infection_type):
        # Retrieve data from the database
        data = self.database.get_amr_data(infection_type)
        if not data:
            return "No data available for this infection type."

        # Process the data (e.g., generating a simple report)
        return (f"AMR Report for {infection_type}:\n"
                f"Resistance Rate: {data['resistance_rate']}%\n"
                f"Recommended Antibiotics: {', '.join(data['antibiotics'])}")

# Integration Test Class
class TestAMRToolIntegration(unittest.TestCase):

    def test_retrieve_and_process_data(self):
        # Initialize the mock database
        mock_db = MockDatabase()

        # Initialize the AMR tool with the mock database
        amr_tool = AMRTool(mock_db)

        # Test the integration between data retrieval and processing for MRSA
        result = amr_tool.retrieve_and_process_data('MRSA')
        expected_output = ("AMR Report for MRSA:\n"
                           "Resistance Rate: 90%\n"
                           "Recommended Antibiotics: Vancomycin, Linezolid")
        self.assertEqual(result, expected_output)

        # Test for a non-existent infection type
        result_no_data = amr_tool.retrieve_and_process_data('Unknown')
        self.assertEqual(result_no_data, "No data available for this infection type.")

# Run the integration tests
if __name__ == '__main__':
    unittest.main()
```

Output:

```
.
----------------------------------------------------------------------
Ran 1 test in 0.000s
OK


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Figure [4]: Example based Integration Test by function and database of AMR Predictor PRO in python.

In component-based integration testing, at least one component must be external, such as a separate class, database, or file, providing the requested reply or data.

4. System Testing

- **Goal**: Test the entire AMR Predictor PRO system, ensuring it behaves as specified.
- **Key Activities**:
  - Test end-to-end functionality, from user input (biological samples or data) to final predictions.
  - Conduct performance testing under different load conditions.
  - Ensure accuracy of antimicrobial resistance predictions across multiple environments and scenarios.
  - Include security testing to verify the protection of sensitive biological data.
- **Deliverable**: Fully tested AMR Predictor system with documented results.

Detailed planning for System testing:

1) Test Environment Setup

The test environment simulates the production environment to ensure the accuracy and relevance of the testing. It includes hardware, software, network configurations, databases, and user privileges. (For example, tools like Cypress can simulate such environments, deploying automated tests and simulating user behavior to assess system performance end-to-end.)

2) Create Test Cases

(Creation of test cases further discussed at section 7 within this report.)

3) Create Test Data

Providing test data that includes real-world scenarios, common use cases, and edge cases.

Examples of test data:

- Genetic data in different file formats for upload.
- Clinical data for multiple patients with different medical histories.
- Pre-processed datasets for model training.
- Test data for integration with external databases.

4) Execute Test Cases

Test cases will be executed manually or through automated testing tools such as Postman for API testing and Selenium for UI testing. Both functional and non-functional testing will be performed on every component.

5) Defect Reporting

Any defects found during the testing process will be logged in a defect tracking system with detailed descriptions, steps to reproduce, and severity. Defect Detection and Analysis metrics must be utilized within system test.

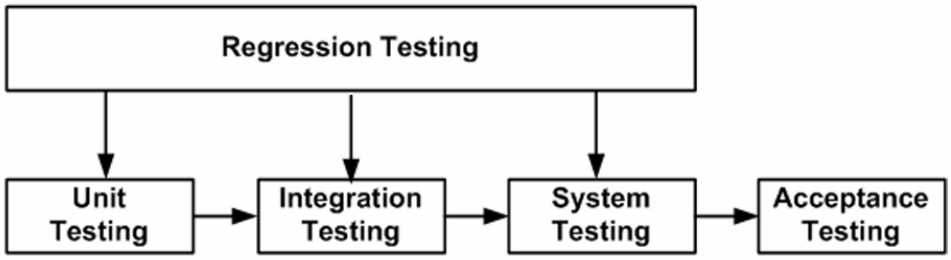
Defect Examples:

- Incorrect rendering of the dashboard for a user role.
- Model training fails with a specific algorithm due to a code bug.
- Real-time data visualization not updating as expected.

6) Regression Testing

Regression testing will be done when flaws are resolved to make sure that other system components continue to function properly and that the fixes do not cause new problems. Automated tests to re-run critical test cases after each update will be used. Focus will be on core modules like Authentication, Model Training, Prediction Interface, and Data Input.

This involves re-running test cases on the core functions like antibiotic recommendation algorithms, data retrieval from the database, and report generation. The focus is on verifying that the tool still delivers accurate antimicrobial resistance predictions, maintains data integrity, and processes user inputs correctly after updates, ensuring no critical workflows are disrupted. Automated regression tests are ideal for this, given the tool's complexity and long-term support requirements.



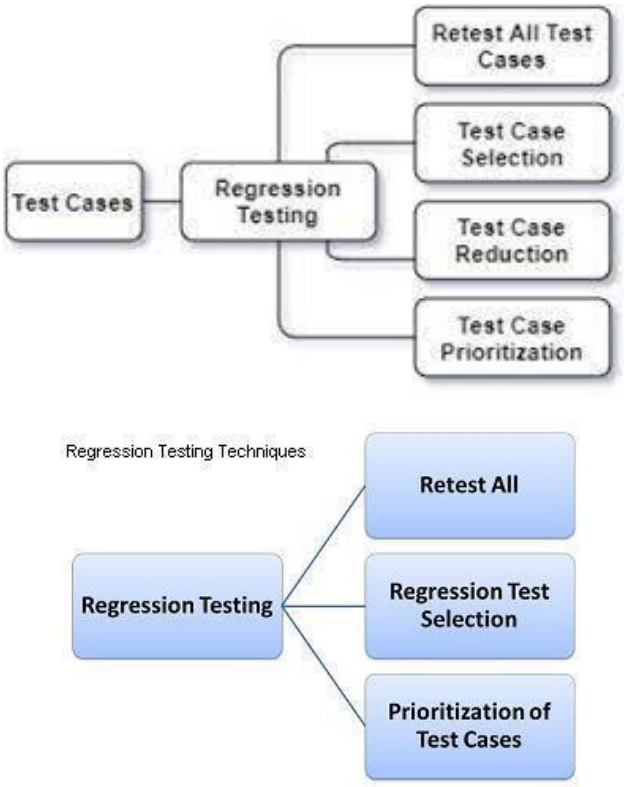


Figure [5]: Examples of regression test diagrams for AMR Predictor PRO.

7) Log Defects

Defects identified during regression testing or retesting will be logged and tracked.

8) Retest

If a test case is unsuccessful, it will be rerun once after fixing the underlying issue to confirm that the problem has been resolved. Retests will also focus on any linked functionality that might have been impacted.

9) Types of System Testing to Perform

To quantify performance tests for the AMR Tool, we need to measure specific metrics such as response time, throughput, concurrent users, and resource utilization (CPU, memory, disk I/O).

Performance Testing:

Response Time: Measure the time it takes for data uploads, model training, and prediction generation. Set quantifiable targets like, for example, < 5 seconds for predictions.

Tools:

- **JMeter**: To simulate real-world user actions and measure response times.

- **Gatling**: Provides detailed performance metrics for web-based systems.

Load Testing:

Concurrent Users: Simulate, for instance, 1,000 to 10,000 concurrent users performing tasks like uploading data or running reports.

Tools:

- **LoadRunner**: Ideal for simulating thousands of virtual users.
- **Apache JMeter**: Widely used for measuring user load and response times.

Stress Testing:

Breaking Points: Push the system to its limits, say 150% of its normal capacity, to determine when it fails or how long it takes to recover.

Tools:

- **Locust**: Open-source tool to stress test the system by gradually increasing load.
- **BlazeMeter**: For simulating high levels of traffic and stress.

Scalability Testing:

Data Volume: Start with 10,000 records and scale up to 1 million records to assess how well the tool handles increased data processing or simultaneous users.

Tools:

- **JMeter** or **Gatling**: To simulate increasing workloads and measure system scalability.

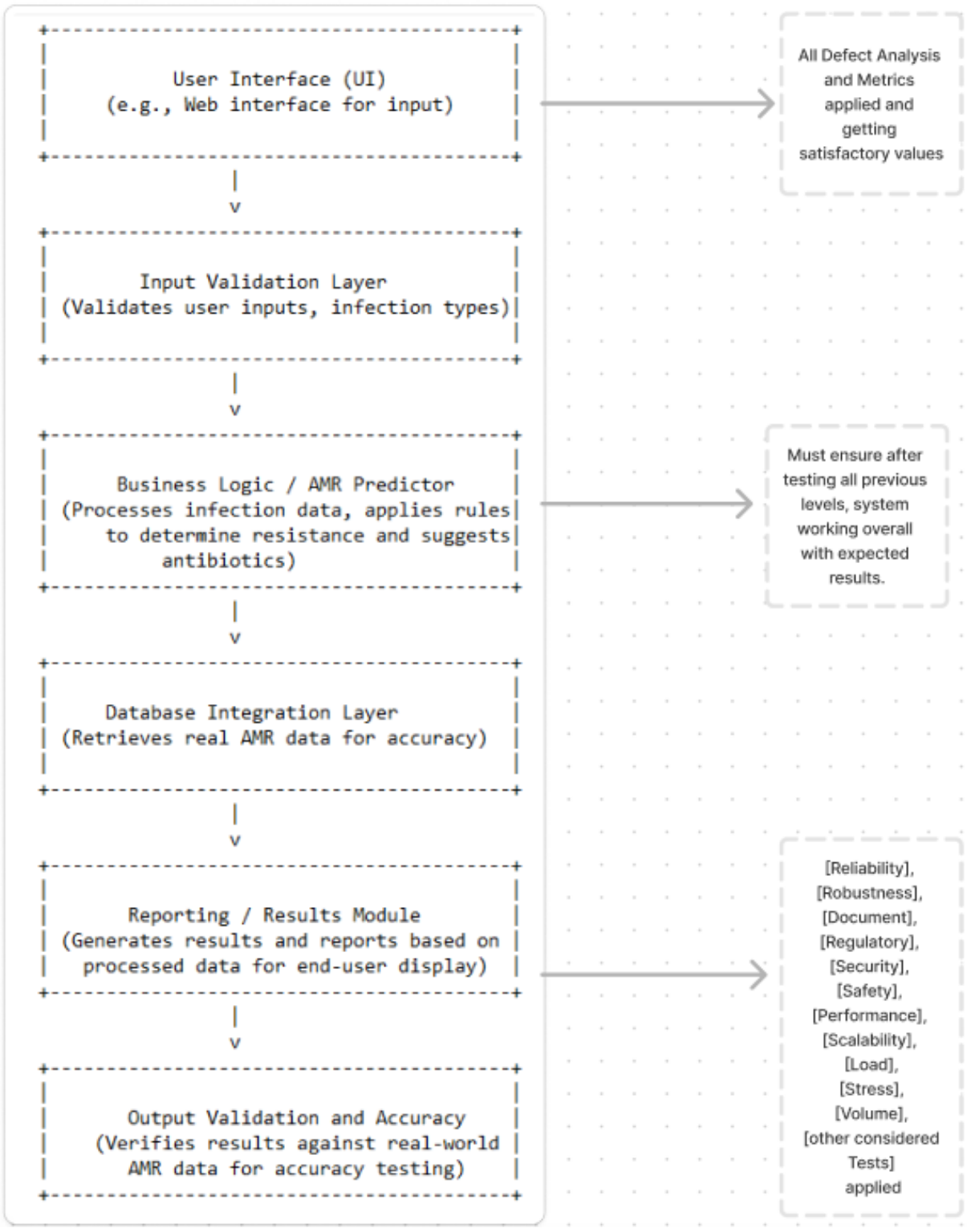


Figure [6]: Example mock diagram of system test for AMR Predictor PRO.

5. Regulatory and Compliance Testing

- **Goal**: Ensure compliance with government regulations and scientific validation standards.
- **Key Activities**:
    - Perform testing to comply with regulatory standards for tools involving healthcare and scientific research (e.g., FDA, EMA, or other global health bodies).
    - Validate the scientific accuracy and integrity of the predictive models.
    - Conduct tests that confirm compliance with privacy laws like HIPAA or GDPR when dealing with patient data.
- **Deliverable**: Regulatory-compliant tool, ready for external audits.

6. User Acceptance Testing (UAT)

- **Goal**: Ensure that the tool meets the expectations of the end users, including scientists, researchers, and medical personnel.
- **Key Activities**:
    - Set up test environments where end users can simulate their workflows.
    - Gather feedback from early adopters such as research institutions, government labs, or hospitals.
    - Conduct tests on user interface and usability, including how results are interpreted by non-technical users.
    - Ensure data visualization and result reporting are intuitive.
- **Deliverable**: Signed-off UAT report and feedback from key stakeholders.

Detailed Planning for Acceptance testing:

Define Acceptance Criteria

- Each component should be tested against specific acceptance criteria based on user stories and requirements. For example:
    - **Authentication Module**: Users should be able to log in with valid credentials and receive appropriate error messages for invalid logins.
    - **Dashboard**: Users should see an overview of recent activities and quick stats without any delays.
    - **Data Input Section**: Users should be able to upload data files and input clinical data without errors and see validation messages for incorrect inputs.

Environment Setup

- Ensure the testing environment is configured to match the production environment as closely as possible.
- Prepare test data for uploading and processing.
- Set up user accounts with different roles (Admin, Researcher, Guest) for role-based testing.

Business acceptance testing (BAT) – Alpha testing

- Assemble a team comprising:
  - Developers: To provide insights on functionality.
  - QA Engineers: To perform systematic testing.
  - Selected End Users: To provide feedback based on their experiences.
- Define a timeline for the alpha testing phase, typically lasting 1-3 weeks.
- Schedule daily or weekly meetings to discuss progress, findings, and any blockers.

User Acceptance Testing (UAT) – Beta testing

- Engage actual users to test the application and gather their feedback on usability and functionality.
- Observe user interactions and identify any pain points or areas for improvement.

Execute Testing

- Perform manual testing of each module against the defined acceptance criteria.
- Document actual results and compare them with expected outcomes.
- Log any defects or issues in a tracking tool.

Final Approval

- Compile a report summarizing the testing outcomes.
- Present findings to stakeholders for review.
- Obtain final sign-off from key stakeholders before the application is moved to production.

Post-Acceptance Actions

- Ensure all critical defects are fixed and retested.
- Plan for regression testing for future updates to the application.
- Set up monitoring and feedback mechanisms post-deployment to catch any issues in the live environment.

7. Special Testing (Anomaly and AI Validation)

- **Goal**: Ensure that AI models used for predictions are validated over time.
- **Key Activities**:
    - Test for anomalous data inputs (e.g., rare resistance patterns) and ensure the tool doesn't generate false predictions.
    - Validate AI models periodically, ensuring predictions stay accurate over time, even as new microbial strains emerge.
    - Perform cross-validation on training models with real-world, unknown datasets.
- **Deliverable**: Ongoing AI validation reports and regular updates to predictive models.

AI is a rapidly evolving field, and the development of antimicrobial resistance analysis and research tools requires constant evaluation of new components and rigorous testing of existing ones. Training data, novel models, and hardware advancements, such as GPUs, must be considered as separate testing components, each with its own associated costs and benefits.

8. Post-Deployment Monitoring and Maintenance Testing

- **Goal**: Ensure continuous operation and apply patches or upgrades as needed.
- **Key Activities**:
    - Conduct regular regression testing after updates or model retraining.
    - Monitor system performance through automated testing suites.
    - Apply security patches and upgrades to meet emerging regulatory needs.
    - Periodically validate model accuracy and adjust based on new research findings.
- **Deliverable**: Continuous operational health and adherence to long-term support contracts.

9. Change Request (CR) and Future Version Testing

Final Phase: Application of Change Requests (CR)

- **Goal**: Adapt the tool to future changes in technology, data, or regulations.
- **Key Activities**:
    - Review all Change Requests based on evolving scientific demands, regulatory shifts, and user feedback.
    - Conduct impact analysis and determine which parts of the system need retesting.
    - Re-perform integration, system, and regulatory testing based on the applied changes.
- **Deliverable**: Revised tool with change management documentation.

Development Model Planning:

For a highly specialized and complex scientific tool like AMR Predictor PRO, which involves predicting antimicrobial resistance, a far better option might be a more holistic approach that integrates both formal verification and model-driven development (MDD) along with scientific validation. MDD focuses on creating abstract models that represent the system's functionality and behavior. This approach is particularly beneficial for complex scientific tools such as AMR Predictor PRO considering precision, Complexity Management, and Integration with Scientific Validation.

Other suggested approaches:

-Risk-Based Testing (RBT)

-Exploratory Testing

-Continuous Integration and Continuous Delivery (CI/CD) with Automated Regression Testing

A combination of suggested approaches with MDD might yield better results for such specialized cases. These approaches were compared with (TDD-Test Driven Development) and (BDD-Behavior-Driven Development) approaches before deciding the plan. TDD is not suited for such special cases.

Test Plan for Training Manuals:

Testing a training manual for a scientific tool like AMR Predictor PRO renowned for its use in antimicrobial resistance (AMR) research and analysis is crucial. The manual must be thoroughly accurate, user-friendly, and scientifically precise, as any errors or ambiguities could lead to improper use of the tool and potentially catastrophic consequences in the research and medical contexts. The effectiveness of such tools often hinges on rigorous validation and testing protocols to ensure their reliability and accuracy in clinical settings.

Given the complexity of AMR prediction and the potential consequences of misdiagnosis, thorough testing and validation of training manuals and predictive models like AMR Predictor Pro are imperative. This ensures that healthcare professionals can rely on these tools for effective decision-making in treating infections.

1. Scientific Content Validation

Objective: Ensure that all scientific concepts, terminology, workflows, and data handling procedures described in the manual are accurate and align with current scientific standards.

Approach:

- **Expert Review:** Engage a panel of domain experts (biologists, microbiologists, pharmacologists, AMR specialists) to review the training manual.
    - They should check for scientific correctness, particularly in areas where the tool analyzes microbial resistance, genomic sequences, and clinical data.
- **Cross-check with Literature**: The content of the manual should be cross-referenced with peer-reviewed scientific papers, textbooks, and industry standards related to antimicrobial resistance.

- **Verification of Algorithms**: Ensure that any descriptions of the predictive algorithms, data analysis techniques, or bioinformatics workflows in the manual are aligned with the actual implementation in the tool.

Deliverables:

- A scientific accuracy report detailing any discrepancies or areas needing revision.

2. User Usability Testing

Objective: Test whether the manual is clear, understandable, and easy to follow for its target users, which may include **researchers**, **scientists**, and **clinical practitioners** with varying levels of technical expertise.

Approach:

- **Personas Creation**: Define key user personas, such as:
  - Experienced researchers familiar with bioinformatics and AMR analysis.
  - Clinical practitioners who may be less familiar with technical details but need the tool for diagnostic purposes.
  - New scientists or graduate students learning about AMR for the first time.
- **Usability Testing**:
  - Provide the manual to a group of users representing each persona and observe their interactions with the tool, using only the manual for guidance.
  - Focus on whether users can easily follow the instructions, understand the concepts, and properly use the tool's features (e.g., entering data, interpreting results).
- **Cognitive Walkthrough**: Have users perform key tasks described in the manual while providing feedback on clarity, ease of understanding, and whether the steps are intuitive.
- **Think-Aloud Protocol**: Ask users to verbalize their thought process as they use the manual and interact with the tool. This helps identify areas where the manual may be unclear or confusing.

Deliverables:

- A usability report highlighting problem areas, user pain points, and suggestions for improving the clarity and flow of the manual.

3. Technical Accuracy Testing

Objective: Ensure that all **technical instructions** (e.g., how to install the tool, configure settings, run analyses, and troubleshoot common issues) are accurate and aligned with the actual functionality of AMR Predictor PRO.

Approach:

- **Hands-On Testing**: Test each step in the manual's technical sections, verifying that the instructions for setup, configuration, and operation match the tool's current version.
    - Check for accuracy in system requirements, software installation, command-line usage, user interface descriptions, and parameter configuration.
- **Version Control Check**: Ensure the manual reflects the latest version of AMR Predictor PRO, accounting for any new features or updates.
- **Reproducibility Testing**: Ensure that users following the instructions can consistently reproduce the expected outputs (e.g., correct resistance predictions, genomic analyses) based on the manual's guidance.

Deliverables:

- A technical accuracy validation report, identifying any inconsistencies or outdated sections in the manual.

4. Compliance with Regulatory Standards

Objective: Ensure that the manual complies with relevant regulatory and ethical standards, particularly if AMR Predictor PRO is being used in clinical or medical research contexts.

Approach:

- **Review for Compliance**: Have legal and regulatory experts review the manual for compliance with:
    - **FDA** or **EMA** guidelines for software used in clinical trials or diagnostic tools.
    - **HIPAA** (Health Insurance Portability and Accountability Act) or **GDPR** for any sections involving patient data or sensitive health information.
    - Ethical guidelines for handling genomic data and AMR research, especially if the tool is being used in healthcare settings.
- **Documentation of Safety Measures**: Ensure the manual clearly outlines safety protocols, ethical guidelines, and the proper use of sensitive data to avoid breaches or misinterpretation.

Deliverables:

- A compliance report confirming that the manual meets all regulatory and ethical standards, or identifying areas needing revision.

5. Educational Effectiveness Evaluation

Objective: Test whether the manual effectively educates users on AMR, bioinformatics, and the tool's specific capabilities. This is critical for users who may not be experts in antimicrobial resistance but need to use the tool for research or analysis.

Approach:

- **Knowledge Retention Testing**: Test the users' knowledge **before and after** using the manual to evaluate how well it conveys important scientific and technical information.
    - Include quizzes, practical exercises, or assessments focused on AMR concepts, tool usage, and interpreting results.
- **Onboarding Effectiveness**: Evaluate how well new users can onboard to the tool and understand core concepts like **genomic data analysis**, **resistance prediction algorithms**, and **data interpretation** based on the manual.
- **Feedback Collection**: Collect qualitative feedback from users on whether they feel the manual has improved their understanding of the tool and AMR concepts. Use surveys, interviews, and focus groups.

Deliverables:

- An educational effectiveness report, including suggestions for improving explanations, diagrams, or supplementary educational materials.

6. Iterative Review and Continuous Improvement

Objective: Make the manual a living document that evolves with the tool over time, ensuring it remains relevant as new features, updates, and scientific discoveries are integrated into AMR Predictor PRO.

Approach:

- **Version Control**: Implement a version control system for the manual to track changes over time. Each new release of AMR Predictor PRO should trigger a review and update of the manual.
- **User Feedback Loops**: Establish feedback mechanisms where users can report unclear instructions or request additional guidance, especially after tool updates.
- **Continuous Improvement Cycles**: Regularly update the manual based on feedback from users, scientific advancements, and new regulations. Consider holding annual reviews or update cycles tied to major tool releases.

Deliverables:

- An updated version of the training manual based on iterative feedback and testing results.

6. TEST CASES/TEST ITEMS

Test Case - 1

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_01	**Test Designed By:**
Test Case Name: Verify Login Functionality	**Test Designed Date:**
Test Priority (Low, Medium, High): High	**Test Executed By:**
Module Name: Login Session	**Test Execution Date:**
Test Scenario ID: TS_101	**Test Case (Pass/Fail):** Pass
Test Scenario: Test the login functionality using valid and invalid credentials	
Pre-condition: The system database must have registered user credentials.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
Check login functionality with a valid username and password	Go to the software enter valid credentials (username and password), and click "Login".	Username: valid_user Password: valid_pass	User can access the dashboard.	User successfully logs in and sees the dashboard.			
Check login functionality with invalid username and password	Enter invalid credentials (username and password), and click "Login".	Username: invalid_user Password: invalid_pass	User cannot access the dashboard.	The system displays an error message: "Invalid credentials."			
Check "Forgot Password"	Click on "Forgot Password" link, submit	Email ID: valid_email@e	User receives a password	A success message appears: "Password			

function ality	email ID for password reset.	xample. com	reset link in their email.	reset link sent."			
Check "Reme mber Me" function ality	Select "Remember Me" checkbox and log in with valid credentials.	Usernam e: valid_us er Passwor d: valid_pa ss	User session is remembere d for future logins.	User stays logged in even after browser restart.			
Check login using card scanner	Insert a valid card for infrared scanning and wait for system validation.	Card: valid_ca rd	User can access the dashboard after scanning.	User successfully logs in and sees the dashboard.			

Test Case - 2

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_02	**Test Designed By:**
Test Case Name: Verify Dashboard Features and Navigation	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**
Module Name: Dashboard	**Test Execution Date:**
Test Scenario ID: TS_102	**Test Case (Pass/Fail):**
Test Scenario: Test various features of the user dashboard, including activity summary, navigation, and profile management.	
Pre-condition: The user must be logged in.	

Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Check dashboa rd summar	Go to the dashboard and view the summary of	None	Summary of activities shown.	Recent activities displayed as per the			

y display	recent activities.			user's interactions.			
2. Check quick stats display	View the quick stats displayed as charts/graphs.	None	Quick stats shown for user's activities.	Stats like model accuracy and data processed are displayed via charts.			
3. Check navigation menu functionality	Expand the navigation menu and access different sections (e.g., Data Input, Predictions, etc.).	None	User navigates to different sections.	The collapsible menu provides access to other sections like AMR Gene Database.			
4. Verify profile management	Click on the user profile and edit details (e.g., name, role).	Name: User Name, Role: Scientist	Profile information updated successfully.	The system allows profile updates and displays the changes.			
5. Check the search bar functionality	Use the search bar to find data within the application.	Search Query: "AMR Data"	The search bar returns relevant data or records.	Search results related to the query are displayed.			
6. Check notifications icon display	View unread notifications from the icon in the header.	New Notification: "Model Training Completed"	User can see unread notifications and take action.	Unread notifications displayed via the notification icon.			

Test Case - 3

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_03	**Test Designed By:**

Test Case Name: Verify Data Upload and Input Functionality	**Test Designed Date:**
Test Priority (Low, Medium, High): High	**Test Executed By:**
Module Name: Data Input	**Test Execution Date:**
Test Scenario ID: TS_103	**Test Case (Pass/Fail):**
Test Scenario: Test the system's ability to handle genetic and clinical data upload and validation.	
Pre-condition: Data files should follow the prescribed format.	

Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Check genetic data upload	Go to the Data Input section and upload a genetic data file.	File: gene_data.fasta	File uploaded successfully and shown in preview.	The system uploads and displays the genetic data file.			
2. Check clinical data input	Fill out the clinical data input form and submit.	Age: 45, Symptoms: fever	Clinical data added to the system.	The clinical data is accepted and stored in the system.			
3. Check preprocessing options	Select preprocessing options (e.g., normalization) before submitting data.	Preprocessing: Normalization	Preprocessing applied to the data	The selected preprocessing is successfully applied.			
4. Check data validation	Upload invalid data and check validation.	Invalid File: incorrect.csv	The system displays validation errors.	The system shows an error message: "Invalid file format."			
5. Check data preview	Preview the uploaded genetic data	File: gene_data.fasta	Preview shows the first few rows of data.	The system displays a preview of the uploaded data.			

Test Case - 4

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_04	**Test Designed By:**
Test Case Name: Verify Model Training and Evaluation	**Test Designed Date:**
Test Priority (Low, Medium, High): High	**Test Executed By:**
Module Name: Model Training	**Test Execution Date:**
Test Scenario ID: TS_104	**Test Case (Pass/Fail):**
Test Scenario: Test the system's ability to train, evaluate, and compare models.	
Pre-condition: Preprocessed data must be available for training.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
1. Check dataset upload for model training	Go to the Model Training section and upload/select dataset.	File: dataset.csv	Dataset uploaded successfully and shown in table format.	The system successfully uploads and displays the dataset.			
2. Check algorithm selection	Select a machine learning algorithm from the dropdown.	Algorithm: Random Forest	The selected algorithm is ready for training.	The system allows selecting and preparing the chosen algorithm.			
3. Configure hyperparameters	Configure hyperparameters for the selected algorithm.	Parameter 1: 100, Parameter 2: 0.1	Hyperparameters configured successfully.	The system accepts the input and configures hyperparameters.			
4. Start model training	Click on "Start Training" and view progress.	None	Model training begins, and the	The system starts training and shows real-			

			progress bar is visible.	time progress updates.			
5. Save the trained model	Click on "Save Model" after training completion.	None	Trained model is saved and listed in a table.q	The system successfully saves the model in the table with download options.			
6. Evaluate the trained model	Click "Evaluate Model" and view performance metrics.	None	Evaluation metrics (Accuracy, Precision, etc.) are displayed.	The system shows model performance metrics in the table.			
7. Compare multiple models	Use the "Model Comparison" feature to compare metrics.	None	Models are compared side-by-side.	The system displays model comparison data (Accuracy, Precision, Recall, etc.).			
8. Check training logs	View logs of actions performed during training.	None	Logs are shown with details of training steps.	The system displays detailed logs of the model training process.			

Test Case - 5

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_05	**Test Designed By:**
Test Case Name: Verify Prediction Functionality with Genetic and Clinical Data	**Test Designed Date:**
Test Priority (Low, Medium, High): High	**Test Executed By:**
Module Name: Prediction Interface	**Test Execution Date:**
Test Scenario ID: TS_105	**Test Case (Pass/Fail):**

Test Scenario: Test the system's ability to input data, generate predictions, and display results.							
Pre-condition: The trained model must be available for prediction.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Input genetic data through the form	Go to the Prediction Interface and fill in the Genetic Data Form fields.	Gene Name: blaOXA, Gene Sequence: ATG...	Genetic data successfully submitted.	The system accepts genetic data and proceeds to the next step.			
2. Input clinical data through the form	Fill out the Clinical Data Form and submit.	Patient ID: 12345, Age: 45, Symptoms: Fever	Clinical data added to the system.	The system accepts clinical data and proceeds to prediction generation.			
3. Generate prediction based on input data	Click "Generate Prediction" and view the Instant Insights section.	None	Predicted antibiotic resistance displayed.	The system generates a prediction and displays resistance, probability, and treatment.			
4. Check confidence score and interval	Review the prediction's confidence score and interval.	None	Confidence score and interval displayed.	The system shows a confidence score and interval next to the prediction result.			
5. Save and export prediction results	Click "Save" and export the results in different formats.	Export format: JSON, CSV, PDF	Prediction results exported successfully.	The system allows the user to export results in the selected formats.			

6. Select drug classes for refining treatment	Use the Drug Class Menu to select specific drug classes.	Class: Beta-Lactams	Drug classes successfully selected for refined prediction	The system refines the prediction and suggests relevant treatments based on the drug class.			
7. Check suggested medicine class	View the Suggested Medicine Class Menu based on the prediction.q	Class: Penicillins	Suggested medicine class displayed based on the prediction.	The system displays suggested medicine classes (e.g., Penicillins, Cephalosporins).			
8. View "Resistant to Drug" table	View the table with genus, species name, and epidemiological data.	None	Data for resistance to drug is displayed in the table.	The system shows the "Resistant to Drug" table with relevant information.			

Test Case - 6

Project Name: AMR Predictor Pro				**Version:** 1.0			
Test Case ID: T_FR_06				**Test Designed By:**			
Test Case Name: Verify Report Generation, Viewing, Export, and Scheduling				**Test Designed Date:**			
Test Priority (Low, Medium, High): Medium				**Test Executed By:**			
Module Name: Reports Section				**Test Execution Date:**			
Test Scenario ID: TS_106				**Test Case (Pass/Fail):**			
Test Scenario: Test the system's ability to generate, view, export, and schedule reports, along with customizations and external database integration.							
Pre-condition: The system must have prediction or analysis data available to generate reports.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**

1. Generate new report	Go to the Reports section, select data, and customize report format.	Data: Model Performance, Prediction Data	New report generated with selected data and format.	The system generates a customized report with selected data and format.			
2. View previously generated report	Go to the View Section and filter reports by date.	Filter: Date: Last Week	Previously generated reports displayed based on filters.	The system displays reports based on the selected date filter.			
3. Export report in PDF format	Select a report and export it as a PDF.	Format: PDF	Report exported successfully in the chosen format.	The system exports the report in PDF format and saves it.			
4. Schedule automated report	Set up automated scheduling for weekly reports.	Schedule: Weekly	Report scheduled for automated generation every week.	The system schedules the report to be generated automatically every week.			
5. Customize report content and layout	Use the Customizability Section to modify the data and layout.	Customization: Layout, Data Selection	Report content and layout customized before generation.	The system allows customization of report content and layout.			
6. View report summary with charts	View the Summary Section to see charts and graphs in the report.	Report: Model Accuracy, Data Analysis	Summary section displays charts and graphs.	The system shows the report summary with charts and graphs.			
7. Share report with other users	Use the Sharing and Collaboration Section to share a report with permissions.	Users: Researcher, Admin	Report shared with specified users.	The system shares the report with the selected users with customized permissions.			

8. Integrate external database data into the report	Select data from an external database for inclusion in the report.	External Database: AMR Gene Database	Data from the external database included in the report.	The system integrates external database data into the generated report.			

Test Case - 7

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_07	**Test Designed By:**
Test Case Name: Verify User Management and Settings Features	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**
Module Name: User Management and Settings	**Test Execution Date:**
Test Scenario ID: TS_107	**Test Case (Pass/Fail):**
Test Scenario: Test the system's user role management, password changes, and notification settings.	
Pre-condition: Admin users must have appropriate privileges.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
1. Manage user roles and permissions	Go to the User Management section, and assign roles and permissions to users.	Role: Researcher, Guest	Users assigned with specific roles and permissions.	The system assigns roles and permissions to users successfully.			
2. Change user password	Go to Settings and change the user's password.	Old Password: old_pass, New Password:	Password updated successfully.	The system allows the user to change their password.			

		new_pass					
3. Enable multi-factor authentication (MFA)	Go to Settings and enable MFA for the user account.	MFA: Enabled	MFA enabled successfully for additional security.	The system enables MFA and prompts users for verification on login.			
4. View user activity logs	Admin user views the activity logs of a specific user.	User: scientist_123	Activity logs displayed for the selected user.	The system shows detailed activity logs for the selected user.			
5. Configure notification preferences	Go to Settings and configure notification preferences for the user.	Notifications: Email, In-app	Notification preferences updated successfully.	The system updates and applies the notification preferences.			
6. Manage data storage and backup	Admin user configures data storage and backup options.	Backup Location: cloud_storage	Backup configuration updated successfully.	The system saves and applies the backup settings.			

Test Case - 8

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_08	**Test Designed By:**
Test Case Name: Verify Help and Support Features	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**
Module Name: Help and Support Interface	**Test Execution Date:**
Test Scenario ID: TS_108	**Test Case (Pass/Fail):**
Test Scenario: Test the help documentation access, support ticket system, live chat, and feedback submission.	

Pre-condition: The user must have access to the system.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Access user manual	Navigate to the Help section and access the user manual.	Help Documentation: User Manual	User can read and browse the user manual.	The system provides access to the full user manual documentation.			
2. Submit a support ticket	Open the Support Ticket section, submit an issue, and track its progress.	Issue: "Error loading data"	Support ticket submitted successfully.	The system confirms ticket submission and allows tracking.			
3. Use live chat support	Start a live chat session with a support agent.	Chat Topic: Data Issue	User engages in real-time chat with an agent.	The system connects the user with a live support agent.			
4. Submit feedback	Go to the feedback form and submit feedback about the application.	Feedback: "Great tool, needs more visualization options"	Feedback submitted successfully.	The system confirms the feedback submission.			

Test Case - 9

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_09	**Test Designed By:**
Test Case Name: Verify Data Analysis and Visualization Features	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**

Module Name: Data Analysis and Visualization Interface					**Test Execution Date:**		
Test Scenario ID: TS_109					**Test Case (Pass/Fail):**		
Test Scenario: Test data visualization tools, filtering, comparison, exporting, and saving custom dashboards.							
Pre-condition: The system must have data to visualize.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Generate heatmap	Go to the visualization section and generate a heatmap.	Data: Antibiotic Resistance	Heatmap generated and displayed.	The system generates a heatmap based on the selected data.			
2. Filter results by pathogen type	Use the filter option to sort data by pathogen type.	Pathogen Type: "MRSA"	Filter applied, and data sorted accordingly.	The system displays only data related to the selected pathogen type.			
3. Compare datasets	Use the comparison tool to compare two datasets.	Dataset 1: AMR Data, Dataset 2: Control Data	Comparison results displayed.	The system compares the datasets and displays the differences.			
4. Export visualization as image	Export the generated heatmap as an image.	Format: PNG	Visualization exported as an image successfully.	The system exports the heatmap as a PNG image.			
5. Save custom dashboard view	Customize the dashboard layout and save it for future use.	Dashboard: Custom Layout 1	Dashboard view saved successfully.	The system saves the custom dashboard layout for future use.			

Test Case - 10

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_10	**Test Designed By:**
Test Case Name: Verify Security and Compliance Features	**Test Designed Date:**
Test Priority (Low, Medium, High): High	**Test Executed By:**
Module Name: Security and Compliance Module	**Test Execution Date:**
Test Scenario ID: TS_110	**Test Case (Pass/Fail):**
Test Scenario: Test data encryption, two-factor authentication, audit logs, and consent management.	
Pre-condition: User data must be handled securely.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
1. Check data encryption	Verify if user data is encrypted during transit and at rest.	Data: User Data, Model Data	User data is encrypted as required.	The system encrypts user data during storage and transmission.			
2. Enable two-factor authentication (2FA)	Go to Settings and enable 2FA for additional security.	MFA: Enabled	2FA enabled successfully for login security.	The system enforces 2FA for enhanced login security.			
3. View audit logs	Admin user views the audit logs for compliance.	Log Type: User Activity	Audit logs displayed for the selected activities.	The system shows detailed logs of user actions for compliance review.			
4. Manage user consent for data usage	Verify if user consent for data usage is tracked and managed.	Consent: Granted, Denied	User consent is managed successfully.	The system tracks and manages user consent for data handling.			

Test Case - 11

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_11	**Test Designed By:**
Test Case Name: Verify External Database Integration Features	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**
Module Name: External Database Integration	**Test Execution Date:**
Test Scenario ID: TS_111	**Test Case (Pass/Fail):**
Test Scenario: Test database connection configuration, data synchronization, and querying.	
Pre-condition: External database access credentials must be available.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
1. Configure database connection	Set up a connection to an external database.	Database: AMR Gene Database	External database connection established.	The system establishes a successful connection with the external database.			
2. Synchronize data with external source	Perform data synchronization with the external database.	Sync Data: Gene Sequences	Data synchronized successfully with the external source.	The system synchronizes data from the external database as required.			
3. Query external database	Query the external database for additional data.	Query: Resistance Gene Data	Query executed, and results displayed.	The system retrieves relevant data from the external database based on the query.			

Test Case - 12

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_12	**Test Designed By:**
Test Case Name: Verify Real-time Data Streaming and Alerts	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**
Module Name: Real-time Data Streaming	**Test Execution Date:**
Test Scenario ID: TS_112	**Test Case (Pass/Fail):**
Test Scenario: Test the system's real-time data streaming and alert features.	
Pre-condition: Live data streams must be available.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
1. Configure real-time data stream	Go to configuration, select internal/external sources for streaming.	Data Source: Bacterial Growth	Data stream configured successfully	The system starts streaming data from the selected sources.			
2. Display live data	Go to real-time data chart and display live data for bacterial growth.	Data: Bacterial Growth	Data displayed in real-time using charts.	The system displays live bacterial growth data in real-time.			
3. Adjust time range on the chart	Use the time range slider to adjust the displayed data range.	Time Range: 12 hours	Time range adjusted successfully.	The system adjusts the chart to show data for the selected time range.			
4. Trigger real-time alerts	Set a threshold for bacterial growth and trigger an alert when exceeded.	Threshold: Growth Rate > 5%	Alert triggered when threshold is exceeded.	The system triggers a real-time alert when the growth rate exceeds			

				the threshold.			
5. Save and export data snapshot	Save a snapshot of the current data and export as PDF.	Format: PDF	Data snapshot saved and exported successfully.	The system saves and exports the data snapshot in the selected format.			

Test Case - 13

Project Name: AMR Predictor Pro				**Version:** 1.0			
Test Case ID: T_FR_13				**Test Designed By:**			
Test Case Name: Verify User Activity Monitoring				**Test Designed Date:**			
Test Priority (Low, Medium, High): High				**Test Executed By:**			
Module Name: User Activity Monitoring				**Test Execution Date:**			
Test Scenario ID: TS_113				**Test Case (Pass/Fail):**			
Test Scenario: Test the system's ability to monitor user activities, sessions, and suspicious behavior.							
Pre-condition: Admin users must have appropriate access rights.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. View user activity logs	Admin user accesses the activity log section.	User: Researcher1	Activity logs displayed for the selected user.	The system displays a log of all activities performed by the user.			
2. Monitor active sessions	Admin views active sessions for all users.	Active Sessions : 2	Active sessions displayed successfully.	The system lists all active sessions with details for each session.			

3. Trigger alert for suspicious activity	Set criteria for suspicious activity and monitor.	Criteria: Multiple failed logins	Alert triggered for suspicious activity.	The system triggers an alert for multiple failed login attempts.			

Test Case - 14

Project Name: AMR Predictor Pro				**Version:** 1.0			
Test Case ID: T_FR_14				**Test Designed By:**			
Test Case Name: Verify Automated Data Backup Features				**Test Designed Date:**			
Test Priority (Low, Medium, High): Medium				**Test Executed By:**			
Module Name: Automated Data Backup				**Test Execution Date:**			
Test Scenario ID: TS_114				**Test Case (Pass/Fail):**			
Test Scenario: Test the system's ability to schedule, perform, and restore data backups.							
Pre-condition: Adequate storage space must be available for backups.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Schedule automated backup	Go to the backup section and schedule a regular backup.	Format: CSV, Frequency: Daily	Backup scheduled successfully.	The system schedules an automated backup at the specified interval.			
2. Perform manual backup	Manually initiate a backup and select the format and location.	Format: Excel, Location: Dropbox	Backup initiated and completed successfully.	The system performs the backup and saves the file to the selected location.			
3. Restore data	Restore data from a previously	Backup: 10/10/2024	Data restored successfull	The system restores the data from			

from backup	completed backup.	Backup File	y from the backup.	the selected backup file.			

Test Case - 15

Project Name: AMR Predictor Pro				**Version:** 1.0			
Test Case ID: T_FR_15				**Test Designed By:**			
Test Case Name: Verify Workflow Management Features				**Test Designed Date:**			
Test Priority (Low, Medium, High): Medium				**Test Executed By:**			
Module Name: Customizable Workflow Management				**Test Execution Date:**			
Test Scenario ID: TS_112				**Test Case (Pass/Fail):**			
Test Scenario: Test the creation, customization, and automation of workflows.							
Pre-condition: Predefined workflow templates should be available.							
Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Create new workflow	Go to the workflow section and create a new workflow.	Workflow: Data Input + Analysis	Workflow created successfully.	The system creates a new workflow combining different modules.			
2. Use workflow template	Select a predefined workflow template and use it.	Template: Prediction Workflow	Template selected and applied successfully.	The system applies the predefined workflow template.			
3. Set automation rules for workflow	Add automation rules to a custom workflow.	Rule: Run Data Input + Analysis	Automation rules applied successfully.	The system applies automation rules to the workflow.			

Test Case - 16

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_16	**Test Designed By:**
Test Case Name: Verify Multi-language Support	**Test Designed Date:**
Test Priority (Low, Medium, High): Low	**Test Executed By:**
Module Name: Multi-language Support	**Test Execution Date:**
Test Scenario ID: TS_116	**Test Case (Pass/Fail):**
Test Scenario: Test the system's ability to switch between multiple languages and manage translations.	
Pre-condition: Translation resources for supported languages must be available.	

Test Case	Step Details	Test Data	Post Condition	Expected Results	Actual Result	Status (Pass /Fail)	Comment
1. Select preferred language	Go to settings, select a language from the dropdown menu.	Language: French	The interface updates to the selected language.	The system successfully switches to the selected language (French).			
2. Admin manages translations	Admin user adds new translations for a language in the settings.	Language: Spanish	Translation resources added successfully.	The system allows the admin to manage translations for the selected language.			
3. Configure localization options	Set date and time format to regional preferences.	Date Format: DD/MM/YYYY	Date and time format updated.	The system updates the date and time format according to the selected localization settings.			

Test Case - 17

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_17	**Test Designed By:**
Test Case Name: Verify Notification Center Features	**Test Designed Date:**
Test Priority (Low, Medium, High): Medium	**Test Executed By:**
Module Name: Notification Center	**Test Execution Date:**
Test Scenario ID: TS_117	**Test Case (Pass/Fail):**
Test Scenario: Test the system's ability to display, mark, and configure notifications.	
Pre-condition: Users must have permission to receive notifications.	

Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. View notification list	Go to the notification center and view the list of notifications.	Notification: "Model Completed"	Notifications displayed in the list.	The system shows all notifications in the notification center.			
2. Mark notifications as read/unread	Mark a notification as read, and another as unread.	Notification: "Data Analysis Done"	Notifications marked accordingly.	The system successfully marks notifications as read or unread.			
3. Configure notification preferences	Go to notification settings and select email as the preferred notification method.	Method: Email	Preferences updated successfully.	The system updates notification preferences as per the user's choice.			

Test Case - 18

Project Name: AMR Predictor Pro	**Version:** 1.0
Test Case ID: T_FR_18	**Test Designed By:**

Test Case Name: Verify API Access and Integration	**Test Designed Date:**
Test Priority (Low, Medium, High): High	**Test Executed By:**
Module Name: API Access and Integration	**Test Execution Date:**
Test Scenario ID: TS_118	**Test Case (Pass/Fail):**
Test Scenario: Test the system's ability to provide secure API access and manage API keys.	
Pre-condition: API access credentials must be configured.	

Test Case	**Step Details**	**Test Data**	**Post Condition**	**Expected Results**	**Actual Result**	**Status (Pass /Fail)**	**Comment**
1. Generate API key	Go to the API management section and generate a new API key.	API Request: GET /data/models	API key generated successfully.	The system generates and displays the new API key.			
2. Manage API key	Go to the API management section and delete an API key.	API Key: key_12345	API key deleted successfully.	The system successfully deletes the selected API key.			
3. Enforce API rate limit	Test the API with multiple requests to enforce rate limits.	Rate Limit: 100 requests/min	API rate limit enforced successfully.	The system enforces the rate limit and rejects requests beyond the limit.			