Final Project Report

**BUDT748 Group 1**

**Client: Trace3**

**Project: Meeting Management Tool**

**Table of Contents**

| 1. **Introduction and Context** |
| --- |
| * 1. **System Overview** |
| * + 1. Core Functionality and Purpose |
| * + 1. Problem Statement and Opportunity Addressed |
| * + 1. Target Audience or Stakeholders |
| * 1. **Purpose** |
| * + 1. Client Needs and Goals |
| * + 1. Expected Outcomes and Impact |
| 1. **The Development Process** |
| * 1. **Steps Taken** |
| * + 1. Research: Problem Identification and Proposed Solution |
| * + 1. Design: Wireframes, Mockups, and Prototypes |
| * + 1. CRUD Matrix for Data Operations |
| * + 1. Development: Tools, Coding, and Testing |
| * + 1. Testing: Feedback Integration and Iterations |
| * + 1. Deployment: Hosting and Accessibility |
| * 1. **Challenges and Solutions** |
| * + 1. Technical, L ogistical, and Design Challenges |
| * + 1. Resolutions and Lessons Learned |
| 1. **System Functionality** |
| * 1. **Core Features and Their Purposes** |
| * 1. **Backend Processes** |
| * 1. **Frontend-Backend Integration** |
| * 1. **Visual Representations (Flowcharts, Diagrams, Tables)** |
| * 1. **Input-Output Walkthrough** |
| * + 1. User Inputs and Outputs |
| * + 1. Core Workflows (e.g., Login, Data Submission, Results Retrieval) |
| * + 1. Error Handling (Invalid Inputs, Feedback Messages) |
| 1. **Standalone Understanding** |
| * 1. **Logical Breakdown of Workflows and User Journeys** |
| * 1. **Visual Representations (Processes and Features)** |
| * 1. **Definitions for Technical Terms and Jargon** |
| * 1. **Feature Documentation** |
| 1. **Deployment Instructions** |
| * 1. **Repository and File Structure** |
| * + 1. GitHub Repository Link |
| * + 1. Description of Directory Organization |
| * 1. **Setup and Installation** |
| * + 1. Required Tools and Software |
| * + 1. Repository Cloning and Dependency Installation |
| * + 1. Configurations (Environment Variables, API Keys) |
| * + 1. Troubleshooting Tips |
| * 1. **Running the System** |
| * + 1. Local Development Server Setup |
| * + 1. Live Server/Cloud Deployment |
| * + 1. Verification Steps |
| 1. **Recommendations for Future Improvements** |
| * 1. **Identified Shortcomings and Stakeholder Feedback** |
| * 1. **Scalability and Performance Enhancements** |
| * 1. **Proposed New Features and Functionalities** |
| * 1. **Development Workflow and Team Collaboration Improvements** |
| 1. **Conclusion** |
| * 1. **Summary of Purpose, Development Journey, and Key Features** |
| * 1. **Impact Achieved for Stakeholders** |
| * 1. **Next Steps** |
| * + 1. System Maintenance or Expansion Plans |
| * + 1. Handover to Client’s Team |
| 1. **Appendices** |
| * 1. **Visual Aids (Diagrams, Flowcharts, Screenshots)** |
| * 1. **Additional Resources or Supporting Documents** |

# **1. Introduction and Context**

## **System Overview**

### **i. Core Functionality and Purpose**

The **Trace3 Meeting Management Tool** is a cutting-edge solution designed to streamline and enhance meeting processes for Trace3’s Strategic Initiatives team. This tool integrates advanced backend technologies with a user-friendly frontend interface to deliver a comprehensive meeting management system. Its core functionalities include automated data collection, real-time stakeholder engagement tracking, and dynamic dashboards that visualize critical Key Performance Indicators (KPIs) such as attendance rates, total meeting hours, and resource allocation metrics. By connecting with platforms like Microsoft Teams and Zoom through APIs, the tool automates data entry, reduces manual overhead, and supports decision-making processes with actionable insights.

The purpose of this project system is as follows:

1. Optimizing Stakeholder Time Commitment
   1. Develop a solution to gather meeting activity on projects tracked by the Strategic Initiatives team.
   2. Avoid burdening stakeholders who have multiple high-priority commitments.
2. Efficient Meeting Data Collection
   1. Design a streamlined system to automate data storage, processing and retrieval for meetings & participants.
3. Visualization of Metrics & Analytics
   1. Dashboard with insightful metrics, KPIs and analytics.
   2. Enabling informed decision-making & efficient project resource scheduling.

### **ii. Problem Statement and Opportunity Addressed**

Trace3’s Strategic Initiatives team faced critical challenges in managing their meetings, such as:

* Manual scheduling and data entry, leading to inefficiencies.
* Limited visibility into stakeholder engagement and meeting outcomes.
* Lack of real-time insights into resource allocation and time distribution.

These challenges resulted in stakeholder overburdening, reduced productivity, and missed opportunities for optimization. Recognizing these inefficiencies, the **Trace3 Meeting Management Tool** was developed as an opportunity to:

1. Replace manual processes with automated systems.
2. Provide real-time data visualization and analytics for better decision-making.
3. Enhance collaboration by integrating existing platforms like Microsoft Teams and Zoom.

This tool transforms meeting management into a streamlined process, enabling Trace3 to address inefficiencies proactively and improve overall project outcomes.

### **iii. Target Audience or Stakeholders**

The primary stakeholders for this tool include:

* **Strategic Initiatives Team at Trace3**: The primary users responsible for managing large-scale projects and coordinating meetings.
* **Stakeholders and Participants**: Individuals involved in meetings, whose time commitments are tracked and optimized.
* **Project Managers**: Users who rely on the tool to visualize meeting KPIs and allocate resources effectively.
* **Trace3 IT Team**: Responsible for hosting, maintaining, and scaling the tool within Trace3’s infrastructure.

By addressing the needs of these stakeholders, the tool ensures widespread adoption and usability across Trace3.

## **Purpose**

### **i. Client Needs and Goals**

The development of the **Trace3 Meeting Management Tool** was driven by key client needs:

1. **Automation**: Eliminate manual tasks such as attendance tracking and resource allocation by integrating with third-party APIs (e.g., Zoom, Teams).
2. **Data-Driven Insights**: Provide detailed KPIs and analytics dashboards to inform strategic decisions and optimize meeting schedules.
3. **Scalability**: Build a system that can scale to accommodate multiple projects, stakeholders, and teams across the organization.
4. **Security and Compliance**: Ensure the system adheres to data privacy standards and Trace3’s internal security protocols.

These goals guided the system’s development, ensuring alignment with the Strategic Initiatives team’s operational priorities.

### **ii. Expected Outcomes and Impact**

The expected outcomes of the tool’s implementation include:

1. **Enhanced Efficiency**:
   * Reduce the time spent on administrative tasks such as data entry and meeting organization.
   * Streamline meeting processes with automated workflows.
2. **Optimized Resource Utilization**:
   * Provide insights into stakeholder participation, enabling better time management and equitable task distribution.
   * Avoid overburdening stakeholders while ensuring project needs are met.
3. **Improved Decision-Making**:
   * Equip project managers with real-time dashboards displaying attendance rates, engagement metrics, and project KPIs.
   * Support data-driven resource allocation and scheduling decisions.
4. **Scalability and Adaptability**:
   * Create a foundation for future enhancements, such as advanced AI-driven insights and integration with additional platforms.
   * Scale the system to support larger teams and increased meeting volumes.

The implementation of this tool is expected to transform Trace3’s meeting management processes, delivering measurable improvements in productivity, stakeholder satisfaction, and project success rates.

# 

# **2. The Development Process**

## **Steps Taken**

### **i. Research: Problem Identification and Proposed Solution**

The development process began with in-depth research to identify key inefficiencies in Trace3's current meeting management system. The findings revealed challenges such as manual scheduling, insufficient data analytics, and lack of integration with existing platforms like Microsoft Teams and Zoom. The team collaborated with Trace3’s Strategic Initiatives team to define a solution: a centralized tool that automates meeting management and provides actionable insights through dynamic dashboards.

The proposed solution included features such as:

* Real-time data collection through API integrations.
* Visualization of KPIs like attendance rates and stakeholder engagement.
* Scalable architecture to support future enhancements.

### **ii. Design: Wireframes, Mockups, and Prototypes**

During the design phase, the team created comprehensive wireframes and prototypes to visualize the system's functionality and user interface. Key deliverables included:

* **Data Flow Diagrams (DFDs)**: Illustrated how data flows between APIs, the backend, and the dashboard.
* **Entity-Relationship Diagrams (ERDs)**: Defined relationships between entities such as stakeholders, meetings, and projects.
* **Wireframes**: Focused on user-friendly navigation, dynamic filters, and clear visualization of meeting data.
* **Prototypes**: Allowed stakeholders to review and provide feedback, which guided iterative refinements.

These designs served as the foundation for the system's development and ensured alignment with client expectations.

### **iii. CRUD Matrix for Data Operations**

The tool's functionality was mapped using a CRUD matrix:

* **Create**:
  + Add new meetings, projects, and stakeholders.
  + Log attendance data from API responses.
* **Read**:
  + Fetch historical and real-time data for meetings and stakeholders.
  + Generate KPI reports.
* **Update**:
  + Modify meeting details, such as schedules and participants.
* **Delete**:
  + Remove outdated or erroneous data entries.

This matrix ensured comprehensive data management and alignment with user needs.

### **iv. Development: Tools, Coding, and Testing**

The implementation phase leveraged modern tools and technologies:

* **Backend**:
  + Developed using Django REST Framework for secure and scalable API management.
  + Utilized Celery for task scheduling and asynchronous operations.
  + Integrated APIs for Microsoft Teams and Zoom to automate data collection.
* **Frontend**:
  + Built with React for dynamic and responsive user interfaces.
  + Styled with Tailwind CSS for consistent and visually appealing designs.
* **Version Control**:
  + GitHub was used for collaborative development and version control.

Coding was structured into sprints, ensuring timely completion of deliverables. Regular code reviews and automated testing maintained quality standards.

### **v. Testing: Feedback Integration and Iterations**

The tool underwent rigorous testing to ensure functionality and reliability:

* **Unit Testing**:
  + Validated individual components, such as API endpoints and database models.
* **Integration Testing**:
  + Ensured seamless communication between the frontend and backend.
* **User Acceptance Testing (UAT)**:
  + Stakeholders tested the system and provided feedback on usability and functionality.
  + Key feedback included improving filter options and refining attendance metrics, which were addressed in subsequent iterations.

### **vi. Deployment: Hosting and Accessibility**

The deployment phase focused on scalability and ease of access:

* **Environment Setup**:
  + Dockerized environments ensured consistent deployment across systems.
  + Configurations were tested locally and prepared for Trace3’s server infrastructure.
* **Documentation**:
  + Comprehensive deployment and user guides were created to assist the Trace3 IT team in hosting and maintaining the tool.
* **Future Scalability**:
  + Designed to accommodate additional features and a larger user base, ensuring long-term usability.

## 

## **Challenges and Solutions**

### **i. Technical, Logistical, and Design Challenges**

1. **Data Access Restrictions**:

* **Challenge**: Security policies prevented access to Trace3’s proprietary data.
* **Solution**: Mock data was created using free accounts on Microsoft Teams and Zoom, simulating real-world conditions.

1. **API Limitations**:

* **Challenge**: Limited functionality in Zoom and Teams APIs hindered real-time data retrieval.
* **Solution**: Developed custom API endpoints and leveraged developer programs for enhanced access.

1. **Design Adaptability**:

* **Challenge**: Incorporating late-stage feature requests, such as week-over-week attendance metrics.
* **Solution**: Maintained a flexible design and iterative development process to accommodate evolving requirements.

1. **Deployment Challenges**:

* **Challenge**: Ensuring compatibility with Trace3’s server environment.
* **Solution**: Docker configurations and thorough testing ensured smooth deployment.

### **ii. Resolutions and Lessons Learned**

* **Adaptability**: Transitioning to mock data allowed the project to progress despite data access challenges, demonstrating the value of contingency planning.
* **Collaboration**: Regular feedback loops with Trace3 stakeholders ensured alignment with client needs and prevented scope creep.
* **Scalability**: Designing for scalability from the outset ensured the system could handle future demands without major rework.
* **Documentation**: Comprehensive documentation facilitated seamless knowledge transfer to the Trace3 IT team.

# 

# **3. System Functionality**

### **Core Features and Their Purposes**

The **Trace3 Meeting Management Tool** delivers robust features to enhance meeting management and resource optimization. These features include:

1. **Automated Data Collection**:
   * Fetches meeting data from Microsoft Teams and Zoom APIs, including attendance, timestamps, and participant details.
   * Automatically updates project and stakeholder records, reducing manual effort.
2. **Dynamic Dashboards**:
   * Provides real-time analytics, including KPIs such as total meeting hours, attendance rates, and stakeholder engagement.
   * Users can apply filters to customize views, such as by project status, date range, or stakeholder involvement.
3. **Stakeholder and Project Insights**:
   * Highlights engagement levels for stakeholders, identifying overburdened participants and recommending better time allocation.
   * Offers detailed project views, showcasing the number of meetings held, stakeholders involved, and attendance trends.

These features collectively address inefficiencies in meeting management, enabling data-driven decision-making and enhancing productivity.

### 

### **Backend Processes**

The backend, implemented using Django REST Framework, handles the core logic and data management for the system. Key backend components and processes include:

1. **Data Models**:
   * **Project**: Stores project-level details (e.g., name, department, status).
   * **Meeting**: Captures meeting details, including start/end times and platform-specific identifiers (e.g., Zoom or Teams meeting IDs).
   * **Stakeholder**: Tracks individuals participating in meetings and their association with projects.
   * **MeetingAttendance**: Logs participant join/leave times and calculates time spent in meetings.
2. **API Endpoints**:
   * **Project ViewSet**: Provides KPIs such as the number of meetings and stakeholders per project. Includes endpoints to retrieve total and weekly hours for meetings.
   * **Meeting ViewSet**: Supports filtering meetings by status (e.g., past, future) and stakeholder email.
   * **Stakeholder ViewSet**: Tracks engagement metrics such as attended hours and invited hours.
3. **Task Scheduling**:
   * Celery automates recurring tasks, such as:
     + Fetching past meeting attendance using commands like log\_zoom\_past\_meeting\_attendance.
     + Synchronizing meeting data through APIs with tasks like get\_zoom\_meetings and get\_teams\_meetings.
4. **Authentication**:
   * Implements token-based authentication for secure access to API endpoints.

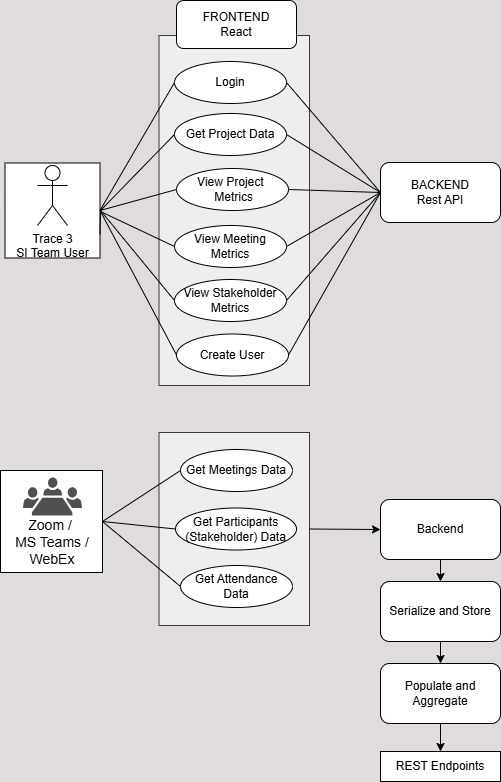
### 

### **Frontend-Backend Integration**

The frontend, built with React, interacts seamlessly with the Django backend to deliver a dynamic and responsive user experience. Key aspects of this integration include:

1. **Data Fetching and Rendering**:
   * Frontend components (e.g., Dashboard.jsx, ProjectOverview.jsx, StakeholderOverview.jsx) fetch data through RESTful APIs.
   * Data is rendered dynamically in tables and charts based on filters and user actions.
2. **State Management**:
   * Tracks filters, selected projects, and user authentication status.
   * Ensures consistent views when navigating between dashboards.
3. **Real-Time Updates**:
   * Automatically updates dashboards with the latest data fetched from the backend.
   * Implements retry mechanisms for API calls to ensure reliability during network issues.

### **Visual Representations**



### 

### **Input-Output Walkthrough**

### **i. User Inputs and Outputs**

**Inputs**:

* Filters for dashboard data (e.g., project status, date range, stakeholder selection).

**Outputs**:

* Projects View
* Stakeholders View
* Upcoming Project Meetings View
* Project-wise Overview for Stakeholder

### **ii. Core Workflows**

**Data Submission**:

* APIs automatically fetch and store meeting data in the backend database.
* Frontend components dynamically request and display this data.

**Results Retrieval**:

* Dashboards display real-time analytics based on user-applied filters.
* Users can retrieve project-level and stakeholder-level insights.

### **iii. Error Handling**

#### **Invalid Inputs**

* The backend employs Django model constraints and validations to ensure data consistency and accuracy. It validates inputs such as meeting data, stakeholder details, and filter parameters.
* The frontend displays user-friendly prompts to guide users in correcting their inputs, ensuring a smooth experience.

#### **API Failures**

* The system leverages Celery to handle API requests. In case of failures, Celery’s retry mechanism ensures consistent data updates by reattempting failed API calls at specified intervals.
* The frontend detects any temporary disruptions and notifies users with clear error messages. It also attempts to refresh data automatically once the issue is resolved.

#### **Feedback Mechanisms**

* The system provides real-time confirmation messages for successful actions, such as applying data filters, generating reports, or updating records.
* This feedback enhances user engagement by informing users that their actions have been successfully completed.

# **4. Standalone Understanding**

## **Logical Breakdown of Workflows and User Journeys**

**1. User Workflows**:

* **Dashboard Navigation**:
  + Users view a high-level overview of meetings, projects, and stakeholder engagement.
  + Filters allow users to customize the data displayed based on project type, date range, or stakeholder involvement.
* **Data Exploration**:
  + Clicking on a project or stakeholder provides detailed insights, such as attendance trends, hours logged, and engagement rates.

**2. System Processes**:

* **Data Collection**:
  + Scheduled tasks in the backend retrieve meeting and attendance data from integrated platforms like Microsoft Teams and Zoom.
  + Data is parsed, validated, and stored in the database for further processing.
* **Data Visualization**:
  + The frontend fetches processed data from backend APIs and renders it as interactive charts, tables, and KPIs.
* **Error Handling**:
  + The system ensures reliability through retry mechanisms for failed API calls and error notifications for users.

## 

## **Visual Representations (Processes and Features)**

**1. Process Flow Diagram**:

* **Data Collection**:
  1. API requests fetch meeting data from third-party platforms.
  2. Backend tasks process and validate the data.
  3. Processed data is stored in relational databases.
* **Frontend Interaction**:
  1. Users interact with dashboards via the frontend.
  2. Frontend components request data from backend APIs.
  3. Responses populate charts, tables, and filters dynamically.\

**2. Feature Table**:

| **Feature** | **Purpose** | **Outcome** |
| --- | --- | --- |
| Dynamic Dashboards | Display KPIs and trends | Improved visibility into meeting analytics |
| API Integrations | Automate meeting and attendance data retrieval | Reduced manual effort |
| Stakeholder Engagement | Track participation metrics | Optimize resource allocation |

**3. User Journey Diagram**:

* **Path 1**: Navigates to Dashboard -> Projects and Stakeholders Tables
* **Path 2**: Clicks on view button for a particular project -> Upcoming Project Meetings
* **Path 3**: Clicks on view button for a particular stakeholder -> Project-wise Overview for Stakeholder

## 

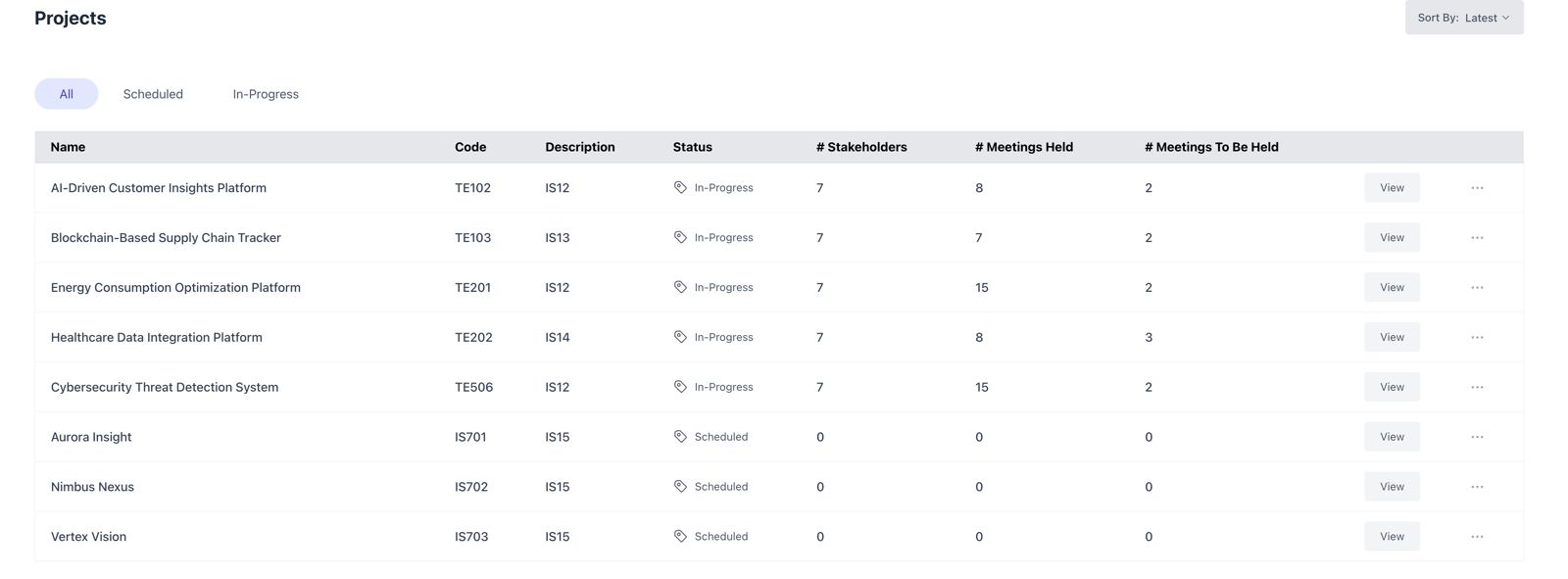
## **Definitions for Technical Terms and Jargon**

* **Django REST Framework (DRF)**: A powerful tool for building web APIs in Python. It provides features like serialization, authentication, and endpoint management.
* **Celery**: An asynchronous task queue used to schedule and execute background tasks, such as fetching data from external APIs.
* **Key Performance Indicators (KPIs)**: Metrics used to evaluate meeting performance, such as attendance rates, hours spent, and engagement scores.
* **API (Application Programming Interface)**: A set of protocols that allow different systems to communicate and exchange data. Examples include Zoom and Microsoft Teams APIs.

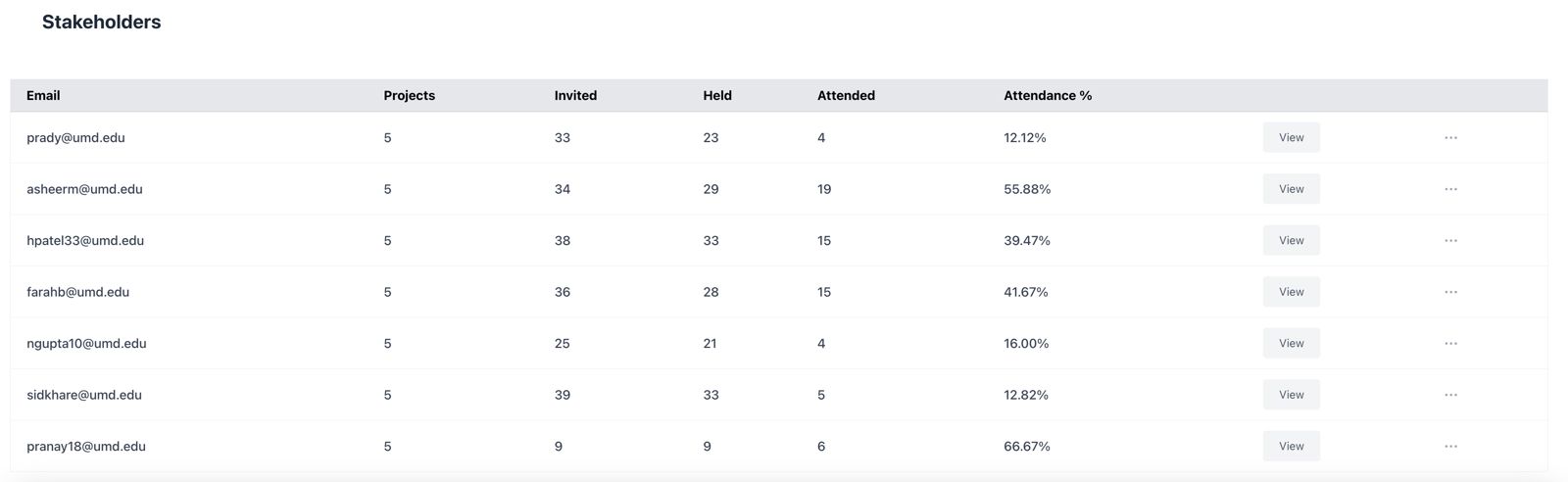
## **Feature Documentation**

Link: [Dashboard Screenshots](https://docs.google.com/document/d/14cEB5WY7pTgR446YEi-mx1tMkvMfk2G7aDt_VBtkpII/edit?usp=sharing)

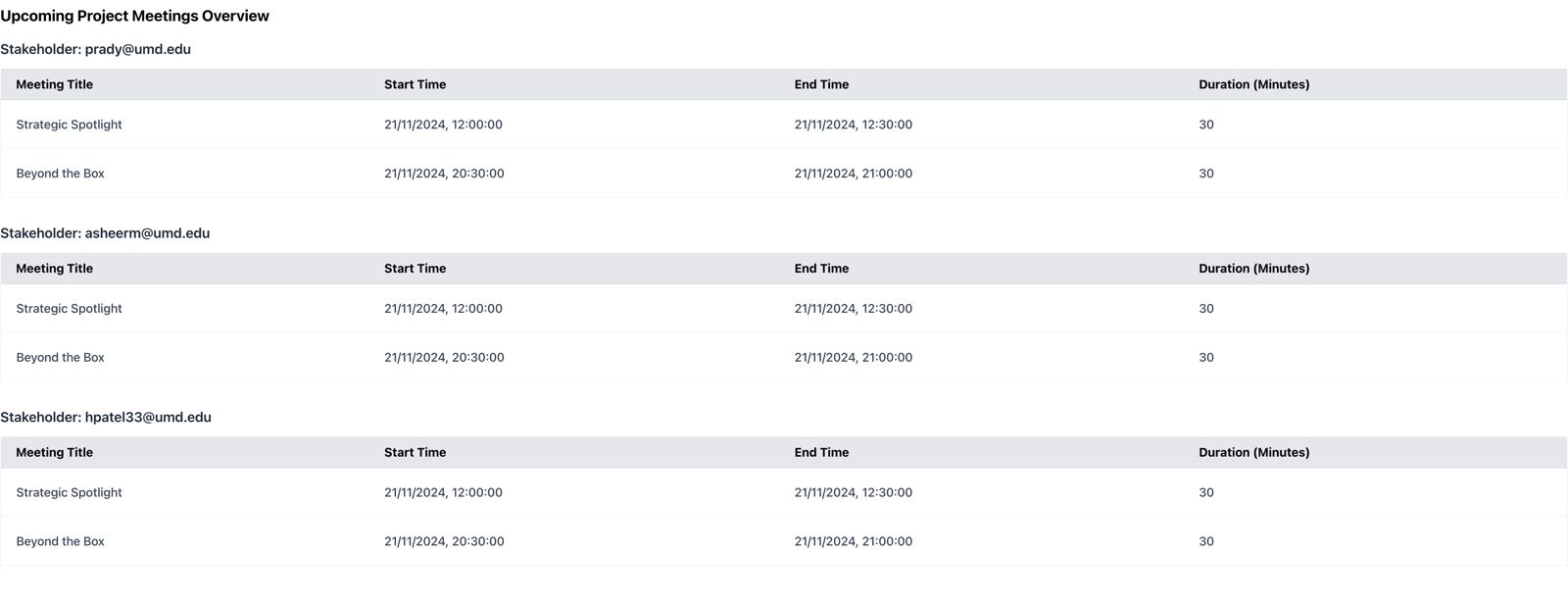
**Projects:**



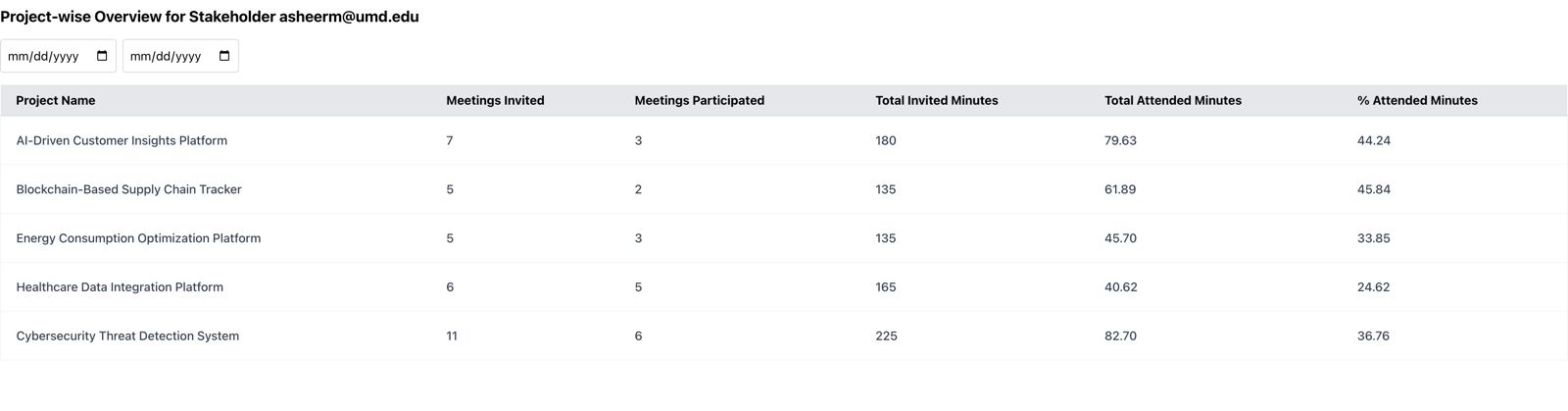
**Stakeholders:**



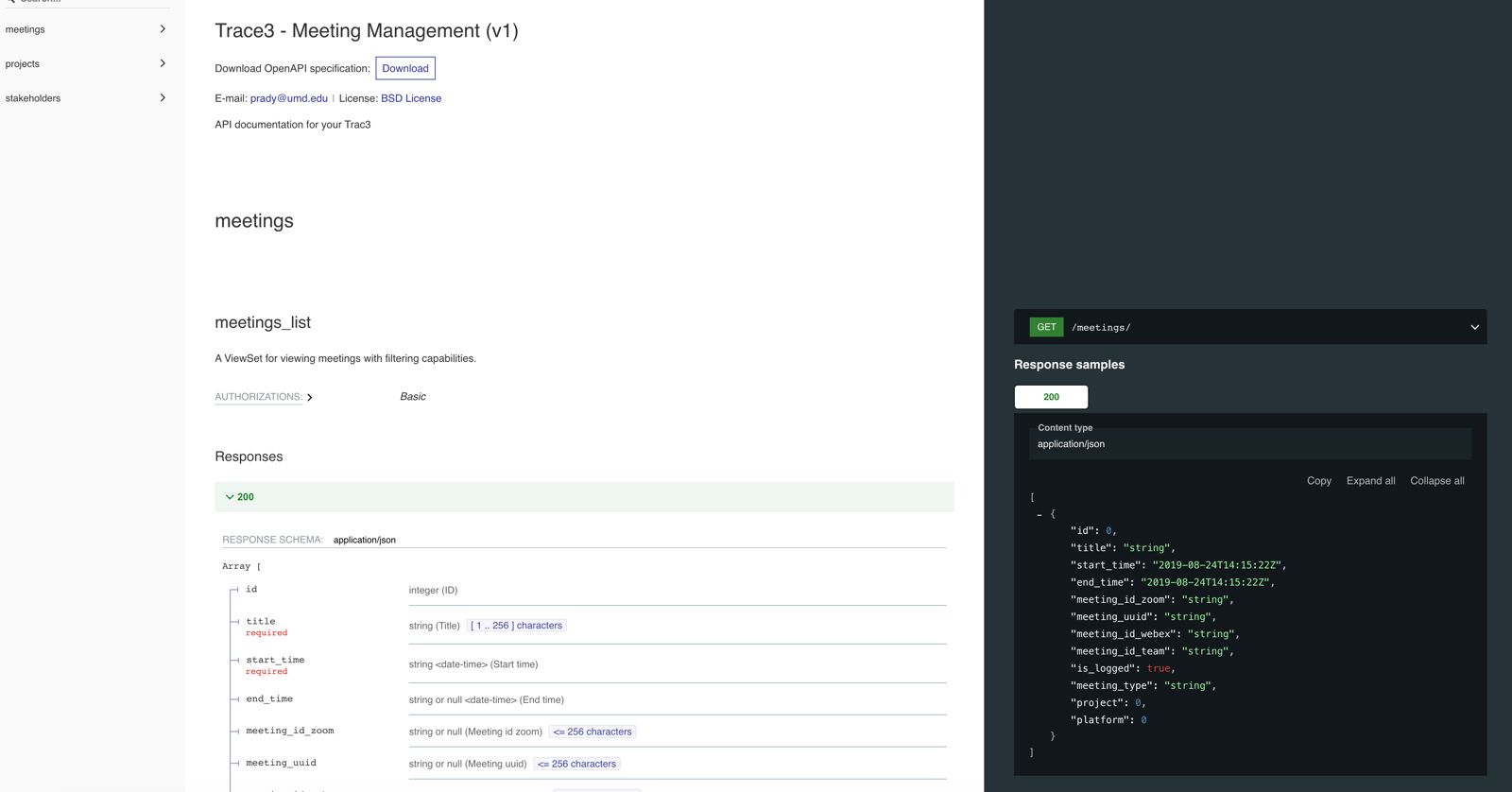
**Upcoming Project Meetings:**

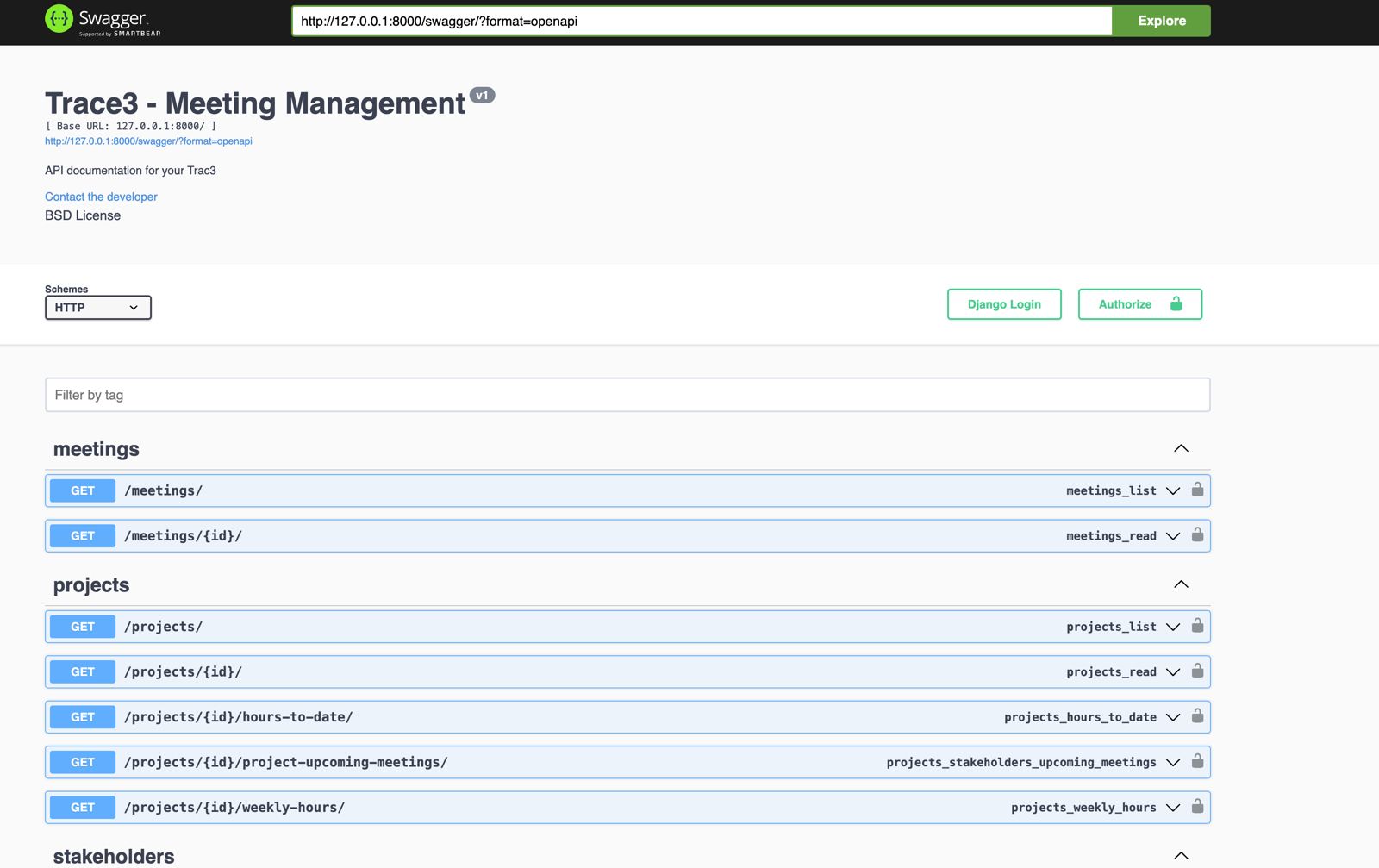


**Project-wise Overview for Stakeholder:**



**Documentation:**





# **5. Deployment Instructions**

## **Repository and File Structure**

### **i. GitHub Repository Link**

Link - <https://github.com/orgs/UMDMSISCapstone/teams/501-trace3-meeting-management-tool>

### **ii. Description of Directory Organization**

The repository is structured to separate frontend, backend, and configuration files for modular development and deployment. Below is an overview of the directory structure:

* **backend/**:
  + Contains Django-based backend logic, including:
    - **meeting\_management/**: Main application with models, views, and serializers.
    - **tasks/**: Celery-based tasks for data synchronization and processing.
    - **settings.py**: Backend configuration file for database, API keys, and middleware.
  + **requirements.txt**: Lists Python dependencies.
* **frontend/**:
  + Contains React-based frontend components, including:
    - **src/**:
      * **Dashboard.jsx**: Main dashboard for visualizing KPIs.
      * **ProjectOverview.jsx** and **StakeholderOverview.jsx**: Components for detailed views.
    - **public/**: Static files, such as index.html for rendering the application.
  + **package.json**: Lists JavaScript dependencies.
* **.gitignore**:
  + Specifies files and directories to exclude from version control.

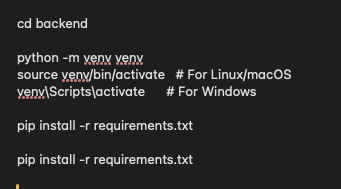
## **Setup and Installation**

### **i. Required Tools and Software**

1. **Backend Requirements:**
   * Python 3.9 or above
   * Django 4.x
   * Celery 5.x
   * Redis (as Celery's message broker)
   * PostgreSQL (preferred database)
2. **Frontend Requirements:**
   * Node.js 16.x or above
   * npm or yarn (for dependency management)
3. **Other Tools:**
   * Docker and Docker Compose (optional for containerized deployment)
   * Git (for repository cloning)
   * IDEs/Editors: VS Code or PyCharm for backend, and VS Code for frontend

### **ii. Repository Cloning and Dependency Installation**

1. Clone the repository
2. Setup the backend



cd backend

python -m venv venv

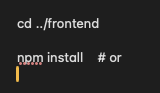
source venv/bin/activate #For Linux/macOS

venv\Scripts\activate #For Windows

pip install -r requirements.txt

pip install -r requirements.txt

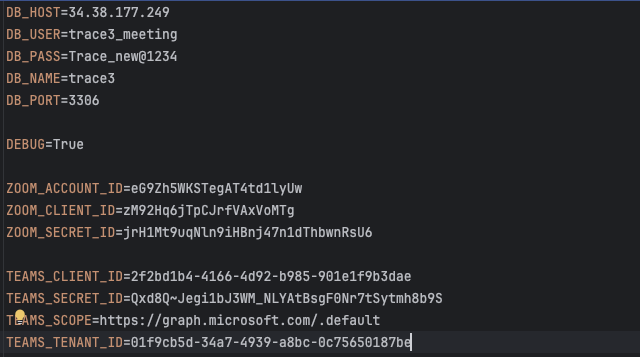
1. Setup the frontend



### **iii. Configurations (Environment Variables, API Keys)**

Environment Variables

Create a .env file with the following params



You can get the zoom api key by creating a zoom app. You can get the teams id from portal.azure.com

### **iv. Troubleshooting Tips**

* <https://www.rootstrap.com/blog/django-best-practices-and-beginner-tips>
* <https://cheatsheetseries.owasp.org/cheatsheets/Django_REST_Framework_Cheat_Sheet.html>
* <https://500apps.com/troubleshooting-rest-api>
* <https://theeventscalendar.com/knowledgebase/troubleshooting-the-zoom-api/>
* <https://central.smarsh.com/s/article/Hub-Article-MSTeams-EAPI-ingestion-slowness-and-other-frequently-seen-issues>

## **Running the System**

### **i. Local Development Server Setup**

Run backend : python3 manage.py runserver

Run frontend : npm run start

### **ii. Live Server/Cloud Deployment**

* Configure Nginx as a reverse proxy and create a Supervisor daemon to manage the application. Ensure the daemon points to the port configured for Uvicorn.
* For frontend deployment, execute npm run build to generate the production build. Copy the contents of the build directory to your server and serve the application using the index.html file.

**iii. Verification Steps**

You can check api at port and use Swagger

https://{BackendURL}/swagger

# 

# **6. Recommendations for Future Improvements**

## **Identified Shortcomings and Stakeholder Feedback**

During development of the platform, there were some challenges. We have explained them below:

* **Challenge**: Limited access to requested meeting data due to security clearance restrictions.
* **Resolution**: Conducted mock meetings on Zoom and Teams to collect real-world data. This ensured the platform was built on real meetings and not mockup data created through an LLM.
* **Challenge**: Did not get access to the development environment of Trace3.
* **Resolution**: We developed the platform on a test environment & used our university’s GitHub repository. We will provide Trace3 with extensive documentation to enable a seamless integration of the tool with their system.
* **Challenge**: Paid organizational accounts for Zoom and Microsoft Teams were required for mock meetings and data collection.
* **Resolution**: Researched and reviewed API documentation for data retrieval and KPIs, and also got paid accounts for both platforms, and got approval from the Microsoft Developer program

**Feedback:**

* We had regular meetings with the client, and conveyed the stages of development, and took feedback. We showcased iterative changes, and improvements to them in subsequent meetings.
* Overall, we received positive feedback from them and they indicated they are looking forward to integrating this platform into their existing systems to track human resources at the company.
* Trace3 must adhere to the established meeting naming conventions to ensure accurate data capture within the tool and we also informed them that we will provide them with comprehensive documentation to assist them in integrating this tool to their existing systems.
* Finally, Ellen at Trace3 has said if required, she will send a feedback note about our team’s effort through email after this week. They were happy about the overall platform.

## **Scalability and Performance Enhancements**

**Database Optimization**:

* **Issue**: As the system scales, database queries may slow down due to increased data volume.
* **Enhancement**: Introduce database indexing for frequently queried fields (e.g., meeting dates, stakeholder IDs). Use caching for high-traffic endpoints like dashboard views.

**Task Management**:

* **Issue**: Current Celery task workers may face bottlenecks under heavy loads.
* **Enhancement**: Migrate to a distributed Celery setup with autoscaling worker nodes to handle increased tasks efficiently.

**Cloud Deployment**:

* **Enhancement**: Transition from local hosting to a fully cloud-native deployment using services like AWS ECS (Elastic Container Service) or Kubernetes. This would improve scalability and fault tolerance.

**Load Balancing**:

* **Enhancement**: Implement a load balancer to distribute requests evenly across backend instances, reducing latency during peak usage.

## **Proposed New Features and Functionalities**

**Advanced Analytics**:

* **Feature**: Introduce AI-powered analytics to predict meeting outcomes based on historical data.
* **Benefits**: Enables better resource allocation and proactive stakeholder engagement.

**Meeting Recommendations**:

* **Feature**: Suggest optimal times for meetings based on stakeholder availability and workload.
* **Benefits**: Minimizes scheduling conflicts and ensures equitable time distribution.

**Expanded Integrations**:

* **Feature**: Add support for platforms like Google Meet and Slack.
* **Benefits**: Broadens the tool’s applicability to other collaboration ecosystems.

**Gamification of Stakeholder Engagement**:

* **Feature**: Implement gamified metrics such as participation scores and badges for active stakeholders.
* **Benefits**: Encourages higher engagement and participation rates.

**Customizable Reports**:

* **Feature**: Allow users to generate tailored reports by selecting specific KPIs, date ranges, or projects.
* **Benefits**: Provides more targeted insights for different stakeholder groups.

## 

## **Development Workflow and Team Collaboration Improvements**

The development workflow and team collaboration during this project were key to its success. Despite minor challenges, the team demonstrated strong technical expertise and effective coordination. Below are the positive highlights of our workflow and collaboration:

* **Role Clarity and Task Distribution**
  + Roles were assigned based on each team member's strengths, ensuring that tasks such as backend development, frontend design, and data testing were handled efficiently.
  + The use of task management tools like Microsoft Planner helped the team stay organized and meet deadlines effectively.
* **Agile Methodology**
  + The team followed an Agile workflow, breaking the project into sprints and conducting reviews after each milestone. This approach ensured steady progress and allowed the team to adapt quickly to new requirements or challenges.
* **Strong Communication**
  + Regular team meetings and use of Microsoft Teams Planner enabled consistent communication and collaboration. Everyone was kept up-to-date on progress, which minimized misunderstandings and streamlined decision-making.
* **Efficient Code Integration**
  + Backend and frontend integration was handled smoothly, thanks to the team’s focus on API documentation and the use of Swagger for API testing. This ensured that all system components worked seamlessly together.
* **Focus on Quality**
  + Peer reviews of code and regular testing using Postman ensured the system met the client’s expectations for functionality and quality. The team’s attention to detail in technical execution was a major strength.
* **Technical Collaboration**
  + The team collaborated effectively on advanced technical tasks, such as creating mock data, integrating APIs (Zoom and Teams), and automating backend processes with Celery. This shared technical effort contributed significantly to the project’s success.

# 

# 

# **7. Conclusion**

**Summary of Purpose, Development Journey, and Key Features**

* Research and Design: Understanding the client's pain points, creating Data Flow Diagrams (DFDs), and designing a seamless workflow.
* Implementation: Building a system using Django REST Framework for backend processing and React for the frontend interface, integrated with APIs for Microsoft Teams and Zoom.
* Testing and Iteration: Generating realistic mock data, simulating real-world scenarios, and refining the system based on feedback.

**Key features include:**

* Automated meeting data collection.
* Dynamic dashboards for visualizing engagement metrics and project-level insights.
* Stakeholder tracking to optimize resource allocation

**Impact Achieved for Stakeholders:**

* Trace3 lacked a system to track human resource allocation to projects or manage meeting-related data.
* The platform provides real-time analytics, helping project managers make data-driven decisions.
* Stakeholders now have better visibility into their time commitments, enabling equitable resource distribution and reducing overburden.

**Next Steps:**

* System Maintenance or Expansion Plans:
  + Explore additional integrations with platforms like Google Meet and Slack.
  + Implement advanced analytics features, such as predictive modeling for stakeholder availability and meeting effectiveness.
  + Optimize database performance to handle larger datasets and improve system scalability.
* Handover to Client’s Team:
  + Provide comprehensive documentation covering deployment, usage, and troubleshooting.
  + Conduct training sessions for Trace3’s IT team to ensure smooth adoption and maintenance.
  + Offer initial support for system deployment and address any client feedback post-launch.

# **8. Appendix**

**Links to additional resources or supporting documents**

* + **Google Slides Presentation:** <https://docs.google.com/presentation/d/1LAxoe9gT8DmjMkstcCvhNgMqGBELNm1efR0vcC5kXxY/edit?usp=sharing>
  + **GitHub:** <https://github.com/orgs/UMDMSISCapstone/teams/501-trace3-meeting-management-tool>
  + **Presentation Recording:**

<https://umd.zoom.us/rec/share/9Uv7EZ_tx-_k11xUJtPWXqmBlH2HPJ9IkNXia-vKwAXRUEcffWQExCi71FgeiIkm.dVOhEwMDuV8wCp9I?startTime=1733779143000>

Passcode: \*K3Scg$1

* + <https://docs.djangoproject.com/en/5.1/howto/deployment/>
  + <https://www.geeksforgeeks.org/how-do-you-deploy-a-react-application/>
  + <https://docs.djangoproject.com/en/5.1/intro/tutorial01/>
  + <https://community.zoom.com/t5/Zoom-App-Marketplace/How-do-I-get-API-Key-amp-API-Secret/m-p/28307>
  + <https://learn.microsoft.com/en-us/graph/auth/auth-concepts>
  + **PoC Emails:**
    - Prof. Paul Shapiro <pshapiro@umd.edu>
    - Farah Bushra <[farahb@umd.edu](mailto:farahb@umd.edu)>, Asheer Mogal <[asheerm@umd.edu](mailto:asheerm@umd.edu)>, Harshil Deepak Patel <[hpatel33@umd.edu](mailto:hpatel33@umd.edu)>, Pradyum Mohta <[prady@umd.edu](mailto:prady@umd.edu)>, Pranayreddy Anthareddy <[pranay18@umd.edu](mailto:pranay18@umd.edu)>, Siddharth Khare <[sidkhare@umd.edu](mailto:sidkhare@umd.edu)>