Sprint 1: Core Microservice Development

User Story 1: Basic PDF File Handling

As a user, **I want** the microservice to accept and process PDF files **so that** I can submit documents for summarization.

Acceptance Criteria:

- The microservice accepts PDF files via API endpoint.
- The microservice validates that the uploaded file is a PDF.
- If the file is not a PDF, the microservice returns an appropriate error message.
- The PDF content is extracted and prepared for summarization.

User Story 2: GenAI Model Integration

As a developer, **I want** to integrate the GenAI model with the microservice **so that** I can generate summaries of the PDF content.

Acceptance Criteria:

- The GenAI model is integrated into the microservice.
- The microservice can generate a summary of the extracted PDF content using the GenAI model.
- The summary is accurate and reflects the key points of the document.
- The summary is returned in a structured format (e.g., JSON).

Sprint 2: API Enhancement and Error Handling

User Story 3: Enhanced API with Summary Options

As a user, **I want** to specify the length and style of the summary **so that** I can customize the output according to my needs.

Acceptance Criteria:

- The API accepts parameters for summary length (short, medium, long) and style (formal, informal).
- The microservice generates summaries based on the specified parameters.
- The output reflects the requested length and style accurately.
- If no parameters are specified, the microservice uses default settings.

User Story 4: Improved Error Handling

As a user, **I want** the microservice to provide meaningful error messages **so that** I can understand and resolve issues.

Acceptance Criteria:

- The microservice returns clear, actionable error messages for invalid inputs.
- Errors include details such as the cause (e.g., unsupported file type, large file size) and possible solutions.
- Error responses follow a consistent format.

Sprint 3: Performance Optimization and Logging

User Story 5: Performance Optimization

As a developer, **I want** the microservice to handle large PDFs efficiently **so that** it performs well under load.

Acceptance Criteria:

- The microservice processes large PDF files (e.g., 100+ pages) within an acceptable time frame.
- The system scales efficiently under increased load, maintaining response times.
- Benchmarking and performance testing are conducted to ensure stability.

User Story 6: Logging and Monitoring

As a system administrator, **I want** comprehensive logging and monitoring **so that** I can track microservice performance and troubleshoot issues.

Acceptance Criteria:

- The microservice logs key events, including API requests, responses, and errors.
- Logs include timestamps, request details, and error messages.
- Monitoring dashboards provide insights into service health, performance metrics, and error rates.
- Alerts are configured for critical events or failures.

Sprint 4: Security and User Authentication

User Story 7: Secure API Access

As a user, **I want** secure access to the microservice **so that** my data is protected.

Acceptance Criteria:

- API access is secured using API keys or OAuth tokens.
- The microservice validates authentication tokens before processing requests.
- Unauthorized requests are rejected with appropriate error messages.
- Data is transmitted securely over HTTPS.

User Story 8: User Authentication and Authorization

As a user, **I want** role-based access control **so that** I can manage access to the microservice.

Acceptance Criteria:

- The system supports user roles (e.g., admin, user) with different levels of access.
- Admins can manage user accounts, including creating, updating, and deleting users.
- Users have access only to the API endpoints and data relevant to their role.
- Access controls are tested to prevent unauthorized actions.

Sprint 5: Deployment and Scaling

User Story 9: Automated Deployment Pipeline

As a DevOps engineer, **I want** an automated CI/CD pipeline **so that** I can deploy updates to the microservice efficiently.

Acceptance Criteria:

- The CI/CD pipeline automatically builds, tests, and deploys the microservice.
- Docker images are built and pushed to ECR on commit to the main branch.
- The pipeline includes steps for unit testing, integration testing, and security scans.
- Deployment to ECS is automated, with rolling updates to minimize downtime.

User Story 10: Auto-Scaling Configuration

As a DevOps engineer, **I want** the microservice to scale automatically **so that** it can handle variable load efficiently.

Acceptance Criteria:

- Auto-scaling is configured based on CPU and memory utilization thresholds.

- The ECS service scales out when utilization exceeds 50% and scales in when it falls below 30%.
- The system supports a minimum of 3 and a maximum of 30 containers.
- Scaling events are logged, and metrics are monitored to ensure optimal performance.