UNIVERSITÉ Concordia UNIVERSITY

Deliverable - 2

Project Title: Codacity (Automated Code Review System)

Team 9			
Name	**Student ID**	**Github Username**	**Email ID**
Rikin Dipakkumar Chauhan	40269431	rikinchauhan01	rikinchauhan3601@gmail.com
Darsh Patel	40232273	iamdarshpatel	darsh0803@outlook.com
Prachi Kalpeshbhai Patel	40291762	prachipatel488	pa_prac@live.concordia.ca
Krishna Alpeshkumar Patel	40232651	krispatel1001	krishna10012001@outlook.com
Bharti Chhabra	40294202	BhartiChh	bharticon210586@gmail.com

3. Feasibility Study Report:

Technical Feasibility

Evaluation of the Technology Requirements for Codacity
Codacity requires a robust technology infrastructure to deliver automated, real-time code reviews across diverse programming environments. Given the performance, scalability, and security demands of our solution, the following technology stack has been selected based on detailed analysis of each component's capability to meet project requirements.

1. **Software Requirements**:

   Programming Languages: Codacity will be developed using **Python** and **JavaScript**. Python's extensive libraries in machine learning, combined with JavaScript's versatility for the front-end, provide a powerful combination for developing both the analytical core and user interface.

   Machine Learning Frameworks: Codacity will utilize **TensorFlow** for its ML-driven recommendations. TensorFlow is a mature, scalable framework that can handle large datasets and real-time processing.

   Web Framework: **Django** will serve as the web framework, allowing for rapid development and easy integration with machine learning models, while **React** will provide a responsive and intuitive front-end user experience.

2. **Hardware Requirements**:

   Server Configuration: Codacity requires high-performance servers to support concurrent processing of code reviews. We recommend a **multi-core, high-memory server setup** (e.g., 16+ GB RAM, 8+ cores per server) to handle intensive ML model computations and database operations.

   GPU Acceleration: To enable rapid processing for model training, **GPU-accelerated servers** (e.g., NVIDIA Tesla GPUs) will be used in the training phase, which is essential for handling large-scale data in real time.

3. **Network Infrastructure**:

   Cloud Deployment: Codacity will be hosted on a **cloud infrastructure** (AWS or Google Cloud) to ensure scalability, flexibility, and access to a wide range of integrated services, such as load balancing, security configurations, and storage solutions.

   Network Security: For secure data transmission, **TLS/SSL encryption** will be implemented on all network communications. The cloud provider's security infrastructure will also ensure data integrity, access control, and regulatory compliance.

4.**APIs and Third-Party Integrations**:

Code Repository Integrations: Codacity will connect with popular repositories like **GitHub, GitLab, and Bitbucket** via their APIs, allowing automated code checks and seamless CI/CD integration.

CI/CD Pipelines: Codacity will integrate with **Jenkins** and **Travis CI** to ensure continuous testing and deployment support.

Security and Vulnerability APIs: Third-party security APIs, such as **OWASP Dependency-Check**, will be integrated to scan for vulnerabilities in real-time, enhancing security features without additional internal development.

5.**Scalability, Security, and Performance**:

Scalability: Codacity will leverage **Kubernetes** for container orchestration, allowing horizontal scaling to support a growing number of concurrent users and repositories.

Performance Optimization: The use of **caching mechanisms** (e.g., Redis) and **optimized database queries** will minimize response times, ensuring Codacity operates efficiently, even under heavy load.

Security: Codacity will implement **multi-factor authentication** (MFA) and **role-based access control** (RBAC) to protect data integrity and access.

Assessment of Implementation Feasibility

The implementation of Codacity's technology stack is feasible due to the availability of mature, open-source tools, skilled resources, and cloud infrastructure. However, several technical challenges may impact implementation:

Integration Challenges: Integrating with multiple CI/CD and repository platforms requires rigorous API management. To address this, Codacity will leverage **API Gateway** for managing and securing third-party integrations, reducing complexities.

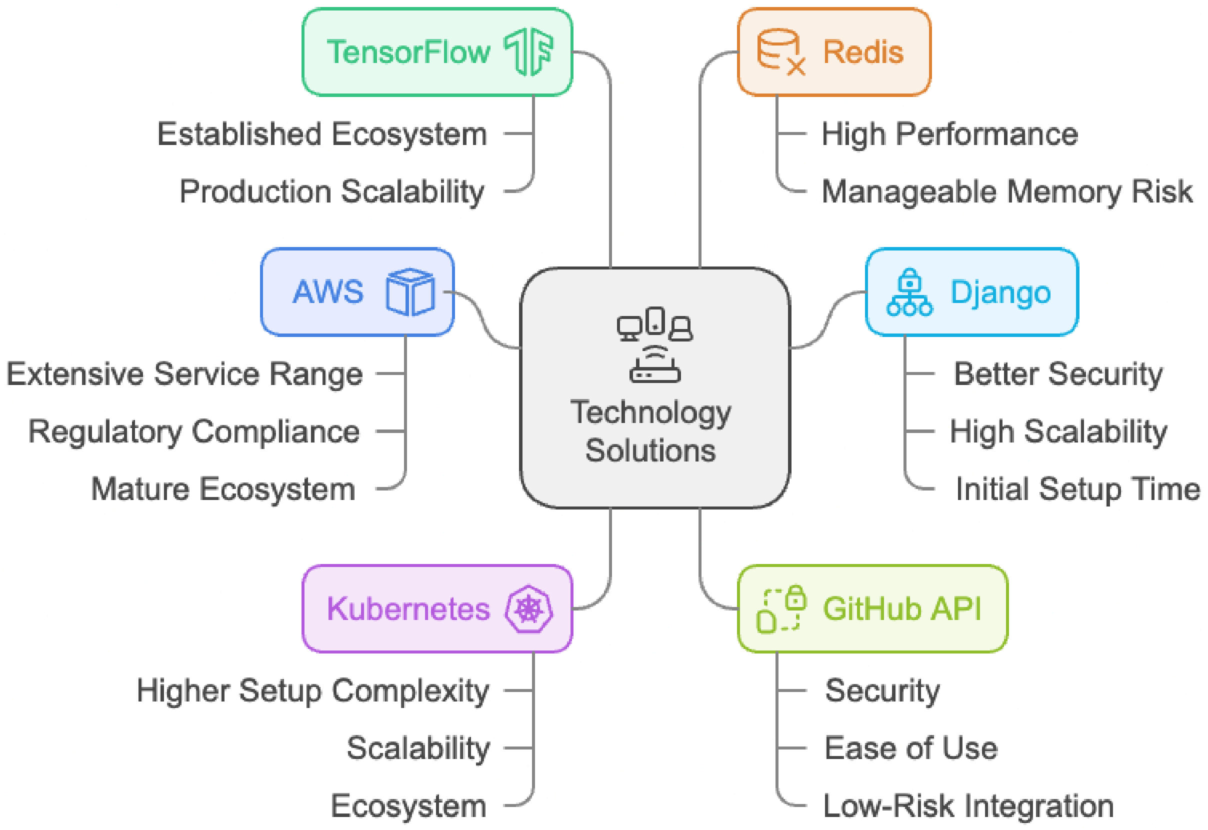
ML Model Training: Real-time code analysis with machine learning requires significant computational power. We will adopt **batch processing** for large datasets and **model optimizations** to reduce training times.

Data Security and Compliance: Given Codacity's application across sensitive industries, data security is paramount. **Continuous monitoring** of compliance with security standards, such as **HIPAA and GDPR**, will be enforced. Codacity will also use **encryption at rest and in transit** to safeguard data integrity.

Future Technological Advancements: Codacity's architecture will remain flexible to adapt to emerging ML frameworks and CI/CD tools, such as MLOps frameworks that streamline deployment and model lifecycle management.

Challenging Component: Technology Comparison Table

Technology Component	**Chosen Option**	**Alternative**	**Pros**	**Cons**	**Risk vs. Reward**
ML Framework	TensorFlow	PyTorch	Well documented, high scalability, strong community support	High memory usage, steep learning curve	High reward for established ecosystem and scalability in production
Cloud Provider	AWS	Google Cloud	Extensive service range, compliance with regulations, mature ecosystem	Vendor lock-in, cost fluctuations	AWS provides reliable scalability and compliance, outweighing the minor risk of cost
Web Framework	Django	Flask	Robust, integrated ORM, strong for web applications	Heavy for small projects, longer learning curve	Django offers better security and scalability, a high reward despite initial setup time
Container	Kubernetes	Docker Swarm	Superior scaling, extensive community, multi-cloud support	Complex initial setup, requires management expertise	Kubernetes' scalability and ecosystem justify its higher setup complexity
Code Repository Integration	GitHub API	Custom Integration	Immediate compatibility, lower development cost, secure	Limited control over updates	GitHub API provides security and ease of use, ensuring quick, low-risk integration
Caching Solution	Redis	Memcached	Fast performance, supports advanced data structures	Requires memory management	Redis offers high reward for performance with manageable memory risk



Selected technology Risk vs Reward visual Representation

Why did we select Python?

Language	Reason Not Chosen	Python Advantage
Java	Complex syntax and longer development times; higher memory usage	Python's simpler syntax speeds up development and reduces code complexity
C++	High performance but lacks built-in ML libraries and has a steeper learning curve	Python's extensive ML libraries (TensorFlow, Scikit-learn) streamline ML-driven features for Codacity
JavaScript	Excellent for front-end but lacks mature ML and data processing libraries on backend	Python is optimized for back-end data processing and ML, which are core to Codacity
Ruby	Slower processing speeds; lacks as many robust ML frameworks	Python offers better performance and a broader range of libraries suited to Codacity's needs.
R	Strong for statistical analysis but not general-purpose or well-suited for web integration	Python's versatility and strong integration support make it ideal for Codacity's multi-functional needs.

Operational Feasibility

Analysis of the Operational Impact of Codacity on Existing Processes

The implementation of Codacity will introduce an automated, streamlined code review system that integrates directly into existing CI/CD workflows. This solution will bring substantial operational changes across several key areas:

1. **Workflow Changes**:

   Code Review Process: Codacity will automate code reviews, reducing reliance on manual reviews and significantly speeding up the process. This will free up developers from spending excessive time on reviews, allowing them to focus more on writing and refining code.

   Feedback Loop: The solution will provide immediate feedback on code quality, eliminating delays in identifying and addressing issues. This will reduce the overall development cycle time and improve deployment frequency.

2. **Roles and Responsibilities**:

   Developers: With Codacity automating code reviews, developers will transition from performing manual reviews to focusing on fixing issues highlighted by the system, improving productivity and minimizing errors.

   DevOps Engineers: Codacity's seamless integration with CI/CD tools will reduce the need for DevOps teams to manually monitor code quality checkpoints, enabling them to allocate more time to optimizing deployment processes.

   Project Managers: Project managers will benefit from the standardized, consistent code quality reports provided by Codacity, enabling more efficient project monitoring and improved delivery timelines.

3. **Effect on Productivity, Training, and User Adoption**:

   Productivity: Codacity will enhance productivity by reducing time spent on reviews and rework due to undetected errors. Additionally, it will ensure consistent quality checks without interrupting the development flow.

   Training Needs: Codacity will require initial training to familiarize users with the system's features, such as the real-time feedback interface, issue categorization, and refactoring suggestions. Training will be straightforward, focusing on practical usage within the CI/CD pipeline, and is expected to have a minimal learning curve for developers familiar with code review tools.

   User Adoption: To promote smooth adoption, Codacity will be introduced gradually, with key features rolled out in phases. This approach allows users to adapt to the automated system incrementally, which will ease the transition and promote a positive reception among team members.

Identification of Potential Challenges and Benefits in the Operational Context

1. **Challenges**:

   User Resistance: Some developers may initially resist the shift to an automated review system due to concerns over a reduced role in quality assurance. To address this, Codacity will emphasize that the tool supports and complements human review by handling repetitive tasks.

   Need for Additional Infrastructure: Codacity requires robust server infrastructure to handle ML-driven processes and integrations with multiple platforms. The existing infrastructure may need scaling, which will be assessed in coordination with the IT team.

   Training and Onboarding: Ensuring all users are comfortable with Codacity's features will require a targeted training program. This will be addressed with hands-on training sessions, focusing on real-world scenarios to facilitate adoption.

2. **Benefits**:

   Increased Efficiency: Codacity's automated, real-time code analysis will significantly improve efficiency, as issues are identified immediately, reducing the time spent on rework.

   Cost Savings: By reducing manual reviews and the number of post-deployment bug fixes, Codacity will decrease both labor and operational costs, contributing to overall project savings.

   Quality Consistency: Codacity's ML-driven analysis ensures consistent application of code quality standards, minimizing human error and ensuring a uniform approach across projects.

   Enhanced Compliance: For industries with stringent regulatory requirements, Codacity's continuous review process will help ensure compliance, reducing the risk of non-compliance penalties.

Challenging Component: Transition Plan and Change Management Strategy

Component	**Proposed Transition Plan**	**Alternative Approach**	**Pros**	**Cons**	**Strategy Comparison**
Training Program	Phased training sessions focused on feature usage in real	Company-wide onboarding workshop	Phased approach allows gradual learning,	Slower full implementatio n due to staggered training	Phased training minimizes user resistance and improves retention

	scenarios				
User Adoption Strategy	Early adopters program to create champions among users	Full rollout to all users simultaneously	Early adopters help set positive tone, build internal advocates	Takes longer for all users to access the system	Early adopter approach supports smoother, gradual adoption
Operational Support	Dedicated Codacity support team for initial 3 months	General IT support	Specialized support ensures quick issue resolution and adoption	Requires additional staffing or reallocation of resources	Dedicated team minimizes initial disruptions and builds user confidence
Feature Rollout Plan	Gradual release of key features, beginning with critical ones	Launch all features at once	Reduces learning curve, increases comfort with initial features	Slower exposure to full system functionality	Gradual rollout minimizes disruptions, encourages smoother adoption
Feedback Collection	Regular user feedback sessions and anonymous surveys	Single survey at the end of initial phase	Ongoing feedback allows timely adjustments	Requires regular scheduling and follow-up	Regular feedback supports agile response to issues and better alignment with user needs

Economic Feasibility

Estimation of the Economic Viability of Codacity

To determine the economic viability of Codacity, a detailed analysis of the short-term and long-term costs has been conducted, covering development, testing, operational expenses, and ongoing maintenance. This comprehensive breakdown ensures that all necessary expenditures, including those for future scaling and potential upgrades, are accounted for:

1. **Development Costs**:

   Personnel: The initial development phase will require a team of software engineers, machine learning specialists, and UI/UX designers. Personnel costs are projected at **$200,000** for the development duration.

   Licensing and Tools: Codacity will rely on specialized tools for machine learning, code integration, and automation testing, including TensorFlow, Jenkins, and API management tools. Licensing and tool costs are estimated at **$15,000**.

   Cloud Infrastructure: The project will initially use cloud services for development and testing, projected at **$5,000/month** for a three-month development period, totaling **$15,000**.

2. **Testing Costs**:

   Quality Assurance: Dedicated quality assurance and testing resources are estimated at **$25,000**. This includes time spent on debugging, user acceptance testing, and security testing.

   User Feedback and Iterative Improvements: Initial user feedback will drive enhancements, requiring an additional **$10,000** in rework and refinements during testing.

3. **Operational Expenses**:

   Hosting and Server Maintenance: Following deployment, Codacity will require ongoing hosting on high-performance servers to support continuous integration and real-time feedback. Estimated at **$3,000/month**, annual hosting costs are **$36,000**.

   Technical Support and Customer Service: A dedicated support team will handle user queries and ensure uptime. Support personnel costs are estimated at **$40,000/year**.

4. **Maintenance and Future Upgrades**:

   Regular Maintenance: Codacity will undergo regular maintenance, including software updates, vulnerability checks, and compatibility enhancements with new CI/CD tools. Yearly maintenance is estimated at **$20,000**.

   Scaling and Upgrades: To handle increased user demand and potential expansions, Codacity will require additional scaling, projected at **$25,000** over the next two years.

Consideration of Resource Availability, ROI, and Cost-Benefit Analysis

Resource Availability:

The development and maintenance teams consist of experienced engineers and QA specialists, ensuring the availability of necessary resources. Given the anticipated growth in Codacity's user base, resource allocation has been planned with scalability in mind, allowing additional personnel to be onboarded if demand increases.

Potential Return on Investment (ROI):

Revenue Streams: Codacity will generate revenue through a subscription-based model targeting development teams and organizations. With monthly subscription fees set at **$200 per team** and anticipated initial adoption of **50 teams**, projected monthly revenue is **$10,000**.

Annual Revenue Projection: Assuming a conservative annual growth of 20% in subscriptions, Codacity's first-year revenue is projected at **$120,000**, with growth to **$144,000** in the second year.

Payback Period: Based on initial development and operational costs totaling **$366,000** for the first year, Codacity is expected to achieve a break-even point in approximately three years as adoption scales.

3. **Cost-Benefit Analysis**:

Cost Savings for Users: By reducing manual review time, Codacity saves an estimated **$2,000 per developer annually** in efficiency gains and reduced rework costs. This cost savings is a major selling point for adoption among development teams, potentially leading to increased user retention.

Long-Term Benefits: Codacity's automated review process will enhance code quality, reduce post-deployment fixes, and improve overall software reliability. These benefits translate into significant cost savings and higher productivity, making Codacity an economically viable investment for development teams.

4. Solution Proposal:

Solution Overview:

Codacity is an Automated Code Review System developed for high-stakes sectors like finance and healthcare, where code quality and security are essential. It leverages AI to streamline code reviews, providing real-time feedback and actionable insights, while integrating seamlessly into CI/CD pipelines. Codacity helps teams maintain high coding standards, reducing review time and improving productivity.

Comprehensive Description of the Proposed Software Solution

1. **Core Functionality**:
    - **Automated Reviews**: Codacity automatically scans code for syntax errors, security vulnerabilities, and quality issues, offering real-time feedback to developers.
    - **Machine Learning Integration**: It learns from past reviews to provide tailored recommendations, enhancing effectiveness over time.
    - **Refactoring Suggestions**: Codacity provides specific suggestions for code improvement, focusing on performance and maintainability.
    - **Security Detection**: It runs comprehensive security checks to identify potential vulnerabilities, ensuring compliance with industry regulations.
2. **System Architecture and Scalability**:
    - **Modular Design**: Codacity's architecture is modular, allowing for easy expansion and feature updates as project needs grow.
    - **Scalability**: Built on a cloud-based infrastructure, Codacity can scale to accommodate projects of any size, from small teams to large enterprises.
    - **Multi-language Support**: It supports multiple programming languages, making it adaptable for diverse teams and projects.
3. **Security and Data Privacy**:
    - **Data Encryption**: Codacity uses encryption to protect data both at rest and in transit, essential for industries dealing with sensitive data.
    - **Compliance Standards**: It adheres to GDPR, HIPAA, and other relevant standards, ensuring regulatory compliance.
    - **Secure Integration**: Codacity uses secure APIs for integration with platforms like GitHub, GitLab, and Jenkins.
4. **Integration and Interoperability**:
    - **CI/CD Pipeline Integration**: Codacity integrates directly into CI/CD workflows, enabling automated reviews as part of the deployment process.
    - **API Flexibility**: Its API allows for custom integrations with other tools, adapting easily to unique organizational needs.
    - **Real-Time Collaboration**: Notifications and shared comments support team-based reviews, aligning with agile and DevOps practices.
5. **Future Enhancements**:
    - **Enhanced AI**: Codacity will continue to improve in identifying complex code issues and recommending architectural improvements.

- **Predictive Analysis and Reporting**: Future updates will include predictive insights and metrics to help teams proactively address potential issues.
- **Expanded Scalability**: With cloud and potential edge computing support, Codacity can evolve with technology, supporting growing and changing teams.

Explanation of How Codacity Addresses the Identified Problem or Opportunity

Codacity directly tackles the challenges of maintaining high code quality, security, and compliance in fast-paced, high-stakes industries like finance and healthcare. In environments where flawed code can lead to security breaches, regulatory non-compliance, and costly post-release fixes, Codacity provides a robust solution that goes beyond traditional manual reviews.

1. **Efficiency and Consistency**:
    - **Problem**: Manual code reviews are time-consuming and can lead to inconsistencies due to human error or varying review standards, slowing development cycles and risking overlooked issues.
    - **Solution**: Codacity automates code reviews, providing consistent, real-time feedback as code is written. This reduces review time by 40-60%, freeing developers to focus on core development tasks while maintaining high standards across the codebase.
2. **Enhanced Security**:
    - **Problem**: Undetected security vulnerabilities can result in data breaches, compliance violations, and financial losses. Manual reviews often miss these subtle vulnerabilities, especially under time constraints.
    - **Solution**: Codacity's AI-driven analysis and in-depth scanning catch security vulnerabilities early in the development cycle. For instance, by identifying issues at the code-writing stage, it helps organizations reduce vulnerabilities by up to 70%, safeguarding against potential breaches and ensuring regulatory compliance.
3. **Improved Code Quality**:
    - **Problem**: Technical debt and inconsistent coding standards are common in fast-paced environments, where enforcing uniform quality is challenging, particularly in large teams.
    - **Solution**: Codacity provides automated suggestions for refactoring, enhancing code readability, maintainability, and scalability. The system learns from historical reviews, adapting its recommendations to align with best practices and team preferences. By maintaining code quality consistently, Codacity reduces technical debt by 20-30% over time.
4. **Scalability and Adaptability**:
    - **Problem**: Existing code review tools are often difficult to scale across diverse teams or projects, limiting their use in growing organizations.
    - **Solution**: Codacity's cloud-based, modular architecture allows it to scale effortlessly with project demands. Supporting multiple languages and providing customizable review rules, Codacity can adapt to various development workflows and project complexities, making it a viable long-term solution for teams of all sizes.
5. **Competitive Advantage and Continuous Improvement**:
    - **Problem**: Traditional code review processes do not adapt or improve over time, limiting their effectiveness and adaptability to evolving coding standards and technologies.

- **Solution**: Codacity's machine learning capabilities enable it to continually refine its analysis based on historical data and feedback, providing increasingly relevant insights. This positions organizations using Codacity at a competitive advantage, with higher code quality, fewer defects per release, and a faster time-to-market.

Key Features and Functionalities

The prioritized list of Codacity's key features and functionalities, organized based on user needs, technical feasibility, and market demand is as follows:

Feature	Priority	Rationale
Automated code review	High	Essential for reducing manual review time and ensuring consistent code quality, a top user is needed in fast-paced environments.
Security Vulnerability Detection	High	Addresses critical security needs in regulated industries, protecting against data breaches and compliance violations.
AI-Driven Insights	High	Provides evolving, intelligent suggestions based on past reviews, meeting market demand for smarter, adaptive solutions.
CI/CD Pipeline Integration	High	Integrates seamlessly into existing workflows, ensuring technical feasibility and high user demand for smooth automation.
Refactoring Suggestions	Medium	Helps users improve code maintainability and reduce technical debt, a valuable but secondary priority after core reviews.
Multi-Language Support	Medium	Supports diverse teams and coding needs, meeting market demand for versatile, cross-language functionality.
Customizable Review Rules	Medium	Enhances flexibility, allowing teams to tailor the tool to their coding standards, which is important for team alignment.
Real-Time Collaboration Tools	Medium	Facilitates team-based reviews with in-app comments and notifications, supporting agile workflows and user collaboration.
Comprehensive Reporting	Low	Provides valuable insights into code quality and performance but can be secondary for users focused on core review tasks.
Scalability and Cloud-Based	Low	Ensures long-term feasibility and growth for

Infrastructure		organizations, important for larger teams but not urgent for all users.

Use Cases or Scenarios Illustrating How Users Will Interact with Codacity

Use Case 1: Developer Submitting Code for Review

- **Scenario**: A developer pushes new code to the repository as part of a feature update.
- **Interaction**: Codacity automatically runs a review on the newly committed code, providing real-time feedback on syntax errors, code smells, and potential vulnerabilities. The developer views the automated feedback, makes adjustments as needed, and pushes the updated code for another round of review.
- **Edge Case**: If the code has an unusual structure or includes deprecated syntax, Codacity flags it, offering recommendations based on best practices. This helps the developer adjust code to current standards, even when handling legacy components.

Use Case 2: DevOps Engineer Integrating Codacity into the CI/CD Pipeline

- **Scenario**: A DevOps engineer is tasked with integrating Codacity into the team's CI/CD pipeline for automated reviews.
- **Interaction**: Codacity is configured to run automatically at each commit and merge step in the CI/CD workflow. The engineer sets rules to halt the deployment process if any critical security vulnerabilities are detected.
- **Edge Case**: If Codacity detects vulnerabilities but they're deemed low-risk or a known issue, the engineer can adjust the review rules to allow non-blocking warnings for that specific vulnerability type, ensuring continuous deployment without compromising security.

Use Case 3: QA Engineer Reviewing Code Quality Reports

- **Scenario**: A QA engineer needs a high-level overview of code quality across different projects for upcoming testing cycles.
- **Interaction**: Codacity generates a report summarizing code quality, security vulnerabilities, and recent refactoring suggestions. The QA engineer reviews the report to prioritize test cases, focusing on areas with detected issues or frequent code changes.
- **Edge Case**: If a report indicates an anomaly, such as a sudden increase in security issues after a specific deployment, the QA engineer flags it for further investigation and collaborates with developers to resolve it before the next testing phase.

Use Case 4: Project Manager Overseeing Development Progress

- **Scenario**: A project manager wants to monitor code quality to ensure project timelines stay on track.
- **Interaction**: Codacity provides the project manager with a dashboard summarizing the number of issues flagged, time saved by automated reviews, and recent improvements in code quality. This helps them make informed decisions about resource allocation and project deadlines.

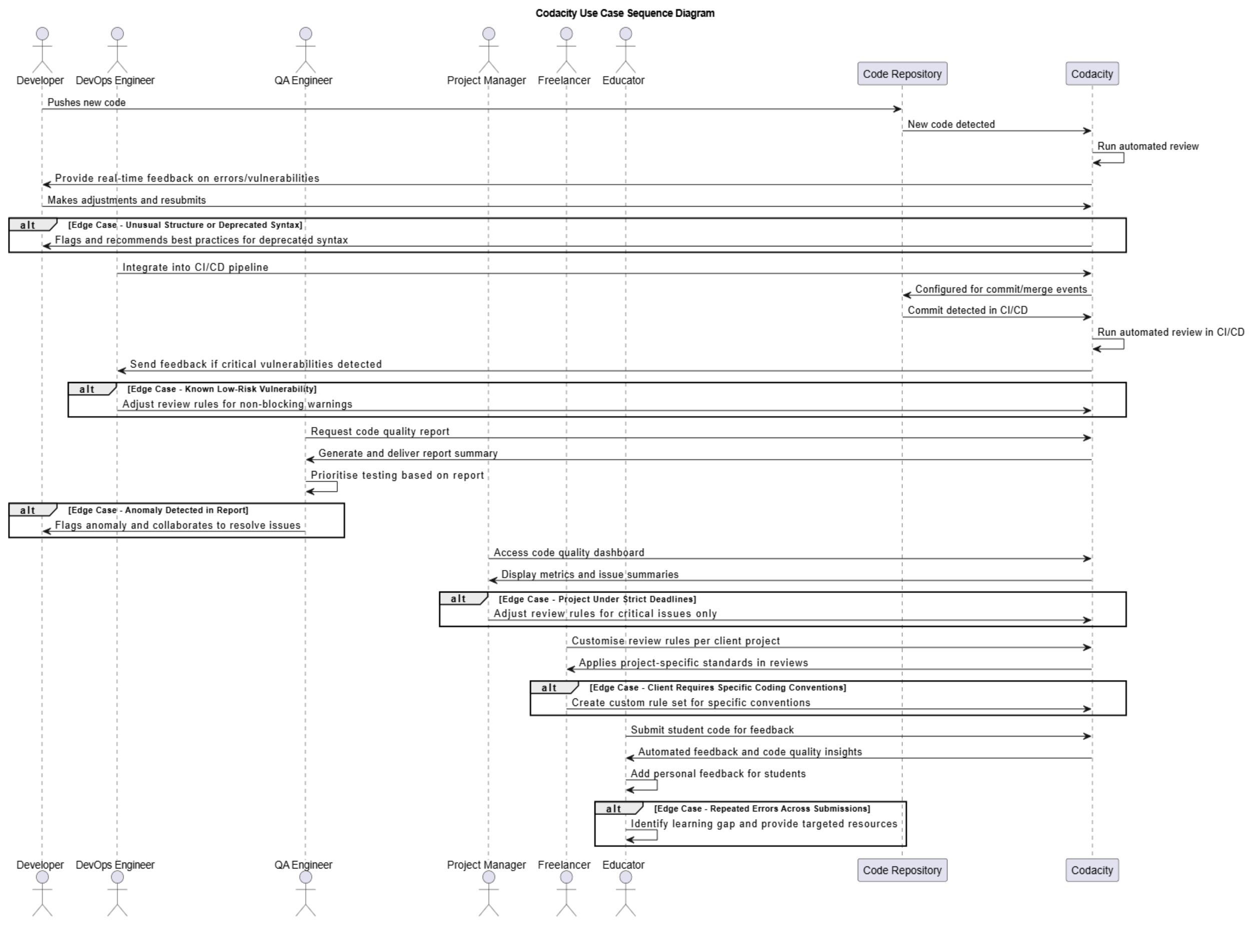
- **Edge Case**: If a project is under strict deadlines, the project manager can adjust Codacity's review rules to focus on critical issues only, prioritizing delivery while still maintaining essential code standards.

Use Case 5: Freelance Developer Working on Multiple Projects with Custom Requirements

- **Scenario**: A freelance developer needs to use Codacity across several client projects with varying code standards and requirements.
- **Interaction**: Codacity's customizable review rules allow the freelancer to set unique standards for each project. The developer can tailor Codacity's checks to meet specific client guidelines and coding practices.
- **Edge Case**: If a client requires specific coding conventions that Codacity doesn't automatically enforce, the developer can create custom rule sets in Codacity, ensuring compatibility with the client's coding guidelines without needing a separate tool.

Use Case 6: Educator Using Codacity for Student Feedback

- **Scenario**: An educator uses Codacity to provide automated code feedback for students learning best coding practices.
- **Interaction**: Codacity reviews each student's code submission, highlighting errors, suggesting improvements, and generating detailed feedback on coding standards. The educator reviews these comments and adds personalized feedback before returning it to the student.
- **Edge Case**: If Codacity detects frequent repeated errors across student submissions, the educator identifies it as a learning gap and can provide targeted guidance or additional resources on that topic.


Codacity Use Case Sequence Diagram
Developer
DevOps Engineer
QA Engineer
Project Manager
Freelancer
Educator
Code Repository
Codacity
Pushes new code
New code detected
Run automated review
Provide real-time feedback on errors/vulnerabilities
Makes adjustments and resubmits
alt
[Edge Case - Unusual Structure or Deprecated Syntax]
Flags and recommends best practices for deprecated syntax
Integrate into CI/CD pipeline
Configured for commit/merge events
Commit detected in CI/CD
Run automated review in CI/CD
Send feedback if critical vulnerabilities detected
alt
[Edge Case - Known Low-Risk Vulnerability]
Adjust review rules for non-blocking warnings
Request code quality report
Generate and deliver report summary
Prioritise testing based on report
alt
[Edge Case - Anomaly Detected in Report]
Flags anomaly and collaborates to resolve issues
Access code quality dashboard
Display metrics and issue summaries
alt
[Edge Case - Project Under Strict Deadlines]
Adjust review rules for critical issues only
Customise review rules per client project
Applies project-specific standards in reviews
alt
[Edge Case - Client Requires Specific Coding Conventions]
Create custom rule set for specific conventions
Submit student code for feedback
Automated feedback and code quality insights
Add personal feedback for students
alt
[Edge Case - Repeated Errors Across Submissions]
Identify learning gap and provide targeted resources
Developer
DevOps Engineer
QA Engineer
Project Manager
Freelancer
Educator
Code Repository
Codacity

Benefits and Impact

Clear articulation of the benefits that users and stakeholder will derive from the solution:

Codacity offers significant advantages for a range of stakeholders, providing immediate and lasting benefits that improve productivity, security, and code quality across teams.

1. **For Developers**:
    - **Short-term**: Codacity's real-time feedback accelerates the review process, allowing developers to quickly address issues and focus more on feature development.
    - **Long-term**: Continuous feedback helps developers improve coding practices, leading to consistently higher code quality and less technical debt over time.
2. **For DevOps Engineers**:
    - **Short-term**: Seamless CI/CD integration ensures that code quality checks are automated within deployment pipelines, reducing the risk of broken builds and last-minute errors.
    - **Long-term**: Codacity's customization options allow DevOps engineers to create a streamlined, scalable pipeline that supports stable, high-quality releases even as the organization grows.

3. **For QA Engineers**:
    - **Short-term**: Codacity's detailed code quality reports enable QA engineers to identify high-risk areas early, improving test focus and efficiency.
    - **Long-term**: By reducing the number of bugs in production, Codacity helps QA teams achieve smoother testing cycles and maintain a higher standard of software quality.
4. **For Project Managers**:
    - **Short-term**: Codacity's dashboard gives managers visibility into code quality metrics, enabling informed decision-making and resource allocation.
    - **Long-term**: Improved code quality and reduced technical debt lead to faster project timelines, cost savings, and fewer post-launch issues, aligning with business goals for on-time delivery and budget efficiency.
5. **For Organisations in Regulated Industries**:
    - **Short-term**: Codacity's security scans provide compliance assurance by detecting vulnerabilities early, helping prevent costly data breaches.
    - **Long-term**: By continuously supporting regulatory standards, Codacity minimizes compliance risks and builds organizational resilience, essential for sectors like finance and healthcare.
6. **Overall Impact**:
    - **Short-term**: Codacity improves workflow efficiency, reduces review time by up to 60%, and cuts down on manual efforts, leading to immediate productivity gains.
    - **Long-term**: Codacity enhances code quality, strengthens security, and lowers long-term costs by reducing technical debt. This solution empowers teams to scale their development processes sustainably, ensuring both stability and growth.

Expected Impact on the Target Audience and the Broader Domain

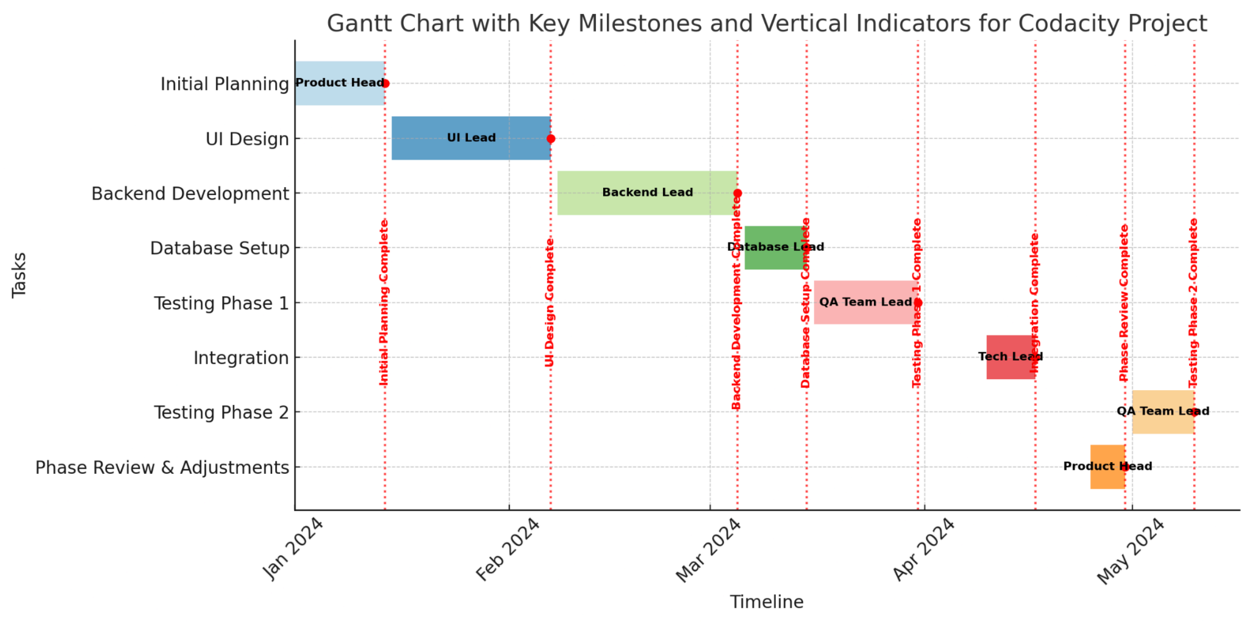
1. **Target Audience Impact**:
    - **Short-term**: Codacity empowers developers, DevOps teams, and QA engineers with real-time insights, reducing code review time and enhancing productivity. Project managers gain better visibility into code quality, improving planning and decision-making.
    - **Long-term**: By improving code quality and reducing technical debt, Codacity enables teams to deliver stable, high-quality software more consistently, building confidence in development processes and lowering post-release maintenance costs.
2. **Broader Industry Impact**:
    - **Short-term**: Codacity's automation of code review processes sets a new standard for efficiency in sectors like finance, healthcare, and tech, where rapid, secure software delivery is critical.
    - **Long-term**: As automated code reviews become industry-standard, Codacity can drive industry-wide shifts towards higher coding standards, reduced vulnerabilities, and a stronger focus on secure, reliable code. This trend contributes to safer applications, lower operational costs, and a more agile response to regulatory requirements.
3. **Societal Impact**:
    - **Short-term**: By minimizing security risks, especially in sensitive domains like healthcare, Codacity indirectly supports data privacy and user trust.

- **Long-term**: Codacity’s focus on automation, efficiency, and quality aligns with broader societal trends towards reliable, responsible tech. This focus can help foster a safer digital ecosystem, benefiting both users and businesses.

5. Project Plan (WBS):

Project Timeline

Gantt Chart or Timeline Illustrating the Key Phases and Milestones:



Allocation of Time to Each Project Phase

The Gantt chart illustrates the project timeline for Codacity, detailing each phase, assigned team members, and key milestones over the six-month development period. Major tasks include:

- **Initial Planning**: Led by the Product Head, completed within the first two weeks of January.
- **UI Design**: Overseen by the UI Lead, spanning late January to early February.
- **Backend Development**: Managed by the Backend Lead, covering February through early March.
- **Database Setup**: Handled by the Database Lead, conducted mid-March.
- **Testing Phases**: Conducted in two parts, led by the QA Team Lead, ensuring functionality and reliability. The first phase begins in late March, and the second in early May.
- **Integration**: Coordinated by the Tech Lead, takes place in mid-April.
- **Phase Review & Adjustments**: Conducted by the Product Head, for project reassessment and adjustments at the end of April.

Milestones and Dilverables

Identification and Description of Major Project Milestones

Initial Planning (Early January)

Lead: Product Head
Description: This phase includes setting project objectives, defining scope, and establishing the initial resource and budget allocation.
Completion Criteria: Documented project charter and scope, initial project plan, and allocated resources are finalized and approved by stakeholders.
Dependencies: This milestone must be completed before moving to UI design, as the foundation and resources defined here impact all subsequent phases.

UI Design (Late January to Early February)
Lead: UI Lead
Description: Creation of the initial user interface design, ensuring it aligns with user requirements and functionality outlined in the planning phase.
Completion Criteria: Approved design prototypes and specifications ready for backend development.
Dependencies: Requires completion of Initial Planning for requirements. UI design completion is essential before backend development can begin.

Backend Development (February to Early March)
Lead: Backend Lead
Description: Development of core backend systems and APIs that support UI functionality and ensure data processing and storage mechanisms are in place.
Completion Criteria: Functional backend system with preliminary integration testing completed, ensuring readiness for the database setup.
Dependencies: UI Design must be finalized. Backend development completion is required before database setup and integration testing.

Database Setup (Mid-March)
Lead: Database Lead
Description: Setup of the database infrastructure, including schema design and initial data population to support backend operations.
Completion Criteria: Database configured, and initial testing completed to validate schema and data accessibility.
Dependencies: Backend Development should be completed to ensure seamless data integration.

Testing Phase 1 (Late March)

Lead: QA Team Lead
Description: The first phase of testing focuses on validating the UI and backend functionalities, identifying bugs, and ensuring system reliability.
Completion Criteria: All critical issues resolved, with sign-off from QA team to proceed to the integration phase.
Dependencies: Requires Database Setup and Backend Development to be fully operational for comprehensive testing.

Integration (Mid-April)

Lead: Tech Lead
Description: Merging different components (UI, backend, database) to form a cohesive system, ensuring that all parts function together seamlessly.
Completion Criteria: Successful integration testing with minimal issues, enabling the start of the final testing phase.
Dependencies: Completion of Testing Phase 1 and all previous development phases to ensure all components are available for integration.

Testing Phase 2 (Early May)

Lead: QA Team Lead
Description: Final testing phase to ensure all functionalities meet quality standards, verifying fixes from the first testing phase and identifying any new issues from integration.
Completion Criteria: All functional and non-functional requirements meet quality standards, with a sign-off for deployment.
Dependencies: Completion of Integration phase to validate the entire system's functionality post-integration.

Phase Review & Adjustments (End of April)

Lead: Product Head
Description: A comprehensive review of project progress, assessing completed milestones, and identifying areas for final adjustments.
Completion Criteria: Approval from all stakeholders on project readiness for final testing and delivery adjustments based on review findings.
Dependencies: All previous milestones need to be completed to assess project performance and final quality.

Work Breakdown Structure (WBS)
The Work Breakdown Structure (WBS) for Codacity’s automated code review system divides the project into five main phases, each with essential tasks for streamlined execution and clear dependencies.

1.**Project Initiation**:

This phase covers requirement gathering, feasibility study, and team setup. Key tasks include defining project objectives, analyzing technical and operational feasibility, and setting up project management tools.

2.**System Design**:

In this phase, the software and architecture are designed, covering back-end architecture, ML model design, and UI/UX development. It includes planning for CI/CD integration and API endpoints for third-party connections.

3.**Development Phase**:

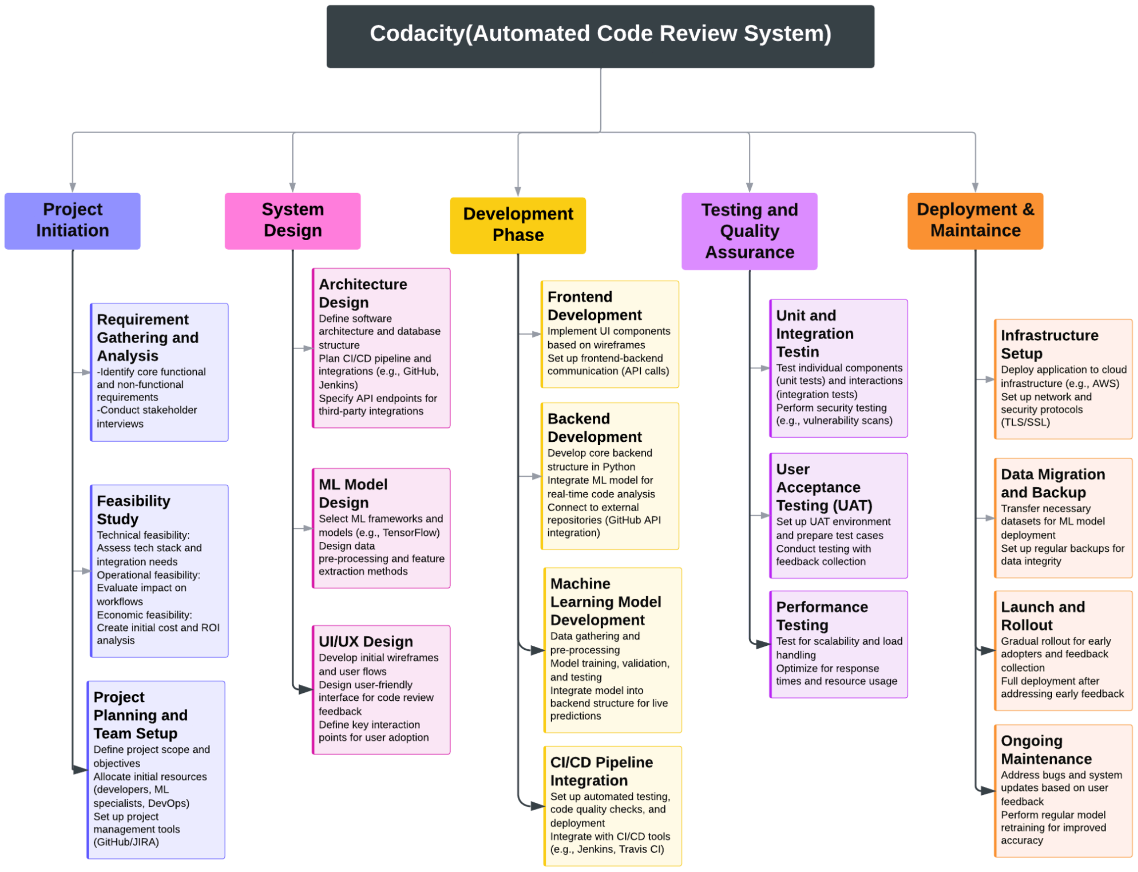
Development tasks are divided into front-end and back-end development, machine learning model implementation, and CI/CD pipeline setup. These components work together to build the functional backbone of Codacity.

4.**Testing and Quality Assurance**:

This phase ensures the system is reliable and efficient through unit, integration, and user acceptance testing (UAT), with a focus on performance optimization.

5.**Deployment & Maintenance**:

The final phase includes infrastructure setup, data migration, and system rollout, followed by ongoing support and maintenance to address user feedback and enable future scaling.



Codacity(Automated Code Review System)

- **Project Initiation**
  - **Requirement Gathering and Analysis**
    -Identify core functional and non-functional requirements
    -Conduct stakeholder interviews
  - **Feasibility Study**
    Technical feasibility: Assess tech stack and integration needs
    Operational feasibility: Evaluate impact on workflows
    Economic feasibility: Create initial cost and ROI analysis
  - **Project Planning and Team Setup**
    Define project scope and objectives
    Allocate initial resources (developers, ML specialists, DevOps)
    Set up project management tools (GitHub/JIRA)
- **System Design**
  - **Architecture Design**
    Define software architecture and database structure
    Plan CI/CD pipeline and integrations (e.g., GitHub, Jenkins)
    Specify API endpoints for third-party integrations
  - **ML Model Design**
    Select ML frameworks and models (e.g., TensorFlow)
    Design data pre-processing and feature extraction methods
  - **UI/UX Design**
    Develop initial wireframes and user flows
    Design user-friendly interface for code review feedback
    Define key interaction points for user adoption
- **Development Phase**
  - **Frontend Development**
    Implement UI components based on wireframes
    Set up frontend-backend communication (API calls)
  - **Backend Development**
    Develop core backend structure in Python
    Integrate ML model for real-time code analysis
    Connect to external repositories (GitHub API integration)
  - **Machine Learning Model Development**
    Data gathering and pre-processing
    Model training, validation, and testing
    Integrate model into backend structure for live predictions
  - **CI/CD Pipeline Integration**
    Set up automated testing, code quality checks, and deployment
    Integrate with CI/CD tools (e.g., Jenkins, Travis CI)
- **Testing and Quality Assurance**
  - **Unit and Integration Testin**
    Test individual components (unit tests) and interactions (integration tests)
    Perform security testing (e.g., vulnerability scans)
  - **User Acceptance Testing (UAT)**
    Set up UAT environment and prepare test cases
    Conduct testing with feedback collection
  - **Performance Testing**
    Test for scalability and load handling
    Optimize for response times and resource usage
- **Deployment & Maintaince**
  - **Infrastructure Setup**
    Deploy application to cloud infrastructure (e.g., AWS)
    Set up network and security protocols (TLS/SSL)
  - **Data Migration and Backup**
    Transfer necessary datasets for ML model deployment
    Set up regular backups for data integrity
  - **Launch and Rollout**
    Gradual rollout for early adopters and feedback collection
    Full deployment after addressing early feedback
  - **Ongoing Maintenance**
    Address bugs and system updates based on user feedback
    Perform regular model retraining for improved accuracy

WBS planning chart

Service Level Agreements (SLAs)

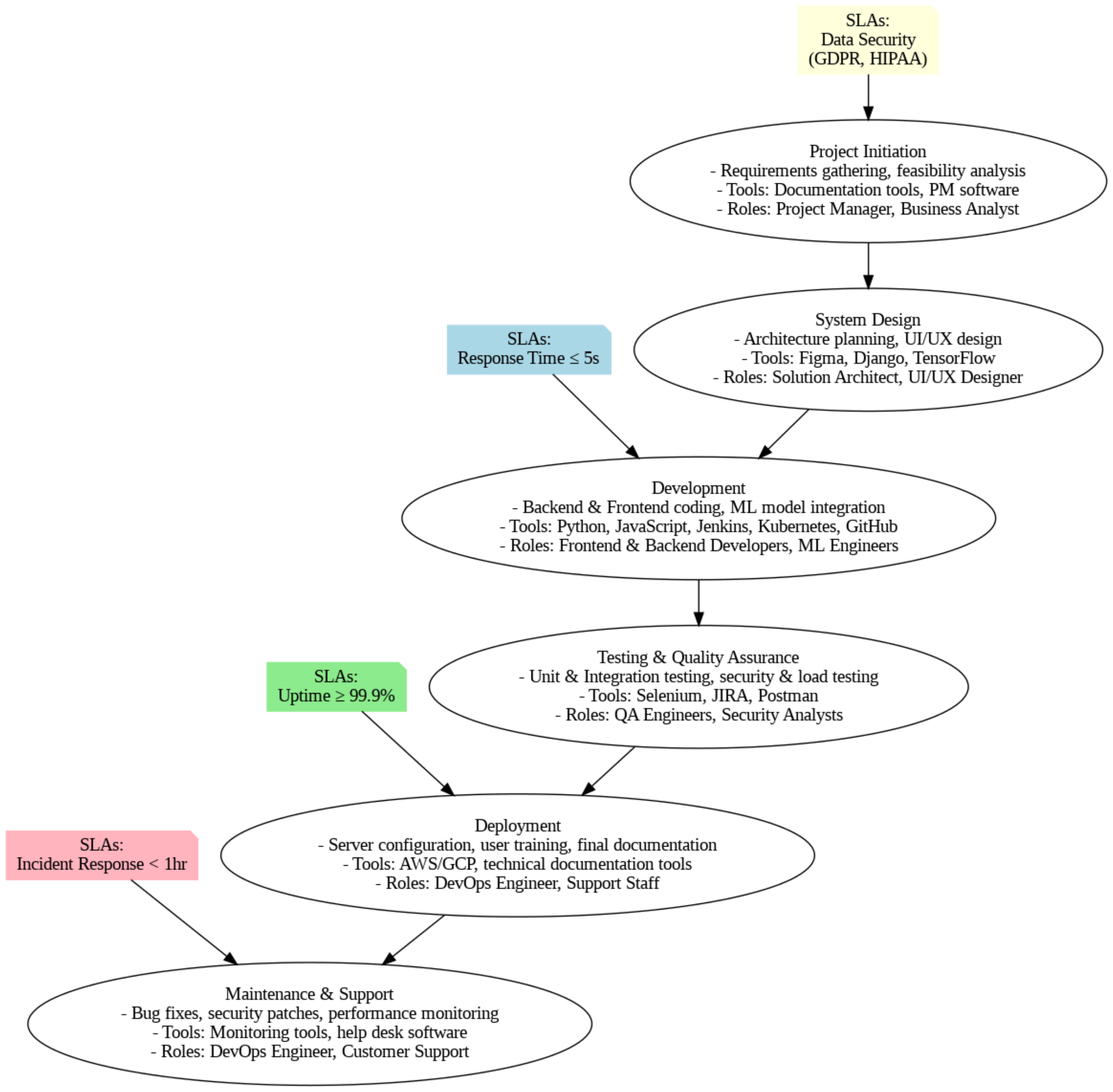
To ensure the reliability, security, and efficiency of Codacity's Automated Code Review System, the following SLAs will be established:

SLA Parameter	Definition	Target Metric
Response Time	Time taken to analyze and provide initial feedback on a code submission.	≤ 5 seconds per code review request.
Uptime	The availability of the Codacity platform, ensuring uninterrupted access to users.	≥ 99.9% uptime.
Data Security	Measures to ensure data encryption, access control, and adherence to regulatory standards.	Compliance with GDPR, HIPAA standards.
Incident Response	Time taken to identify, address, and notify users about any critical issue or security breach.	Initial response within 1 hour; full resolution within 24 hours.

Listing of Deliverables at Each Project Phase (Resources Involved in Each Phase)

Below is a breakdown of the resources involved in each phase of Codacity's development and deployment.

Phase	Key Activities	Roles Involved	Resources
Project Initiation	Requirements gathering, feasibility analysis, stakeholder interviews.	Project Manager, Business Analyst	Documentation tools, PM software.
System Design	Architecture planning, UI/UX design, technical stack selection.	Solution Architect, UI/UX Designer	Figma, Django, TensorFlow.
Development	Frontend and backend coding, ML model integration, CI/CD setup.	Frontend & Backend Developers, ML Engineers	Python, JavaScript, Jenkins, Kubernetes, GitHub.
Testing & Quality Assurance	Unit and integration testing, security and load testing.	QA Engineers, Security Analysts	Selenium, JIRA, Postman.
Deployment	Server configuration, user training, final documentation.	DevOps Engineer, Support Staff	AWS/GCP, technical documentation tools.
Maintenance & Support	Bug fixes, security patches, performance monitoring.	DevOps Engineer, Customer Support	Monitoring software, help desk.


SLAs:
Data Security
(GDPR, HIPAA)
Project Initiation
- Requirements gathering, feasibility analysis
- Tools: Documentation tools, PM software
- Roles: Project Manager, Business Analyst
SLAs:
Response Time ≤ 5s
System Design
- Architecture planning, UI/UX design
- Tools: Figma, Django, TensorFlow
- Roles: Solution Architect, UI/UX Designer
Development
- Backend & Frontend coding, ML model integration
- Tools: Python, JavaScript, Jenkins, Kubernetes, GitHub
- Roles: Frontend & Backend Developers, ML Engineers
SLAs:
Uptime ≥ 99.9%
Testing & Quality Assurance
- Unit & Integration testing, security & load testing
- Tools: Selenium, JIRA, Postman
- Roles: QA Engineers, Security Analysts
SLAs:
Incident Response < 1hr
Deployment
- Server configuration, user training, final documentation
- Tools: AWS/GCP, technical documentation tools
- Roles: DevOps Engineer, Support Staff
Maintenance & Support
- Bug fixes, security patches, performance monitoring
- Tools: Monitoring tools, help desk software
- Roles: DevOps Engineer, Customer Support

Visual Flow of Resources Across Phases

6. Risk Assessment and Mitigation:

Risk Identification

Comprehensive List of Potential Risks Associated with the Project:

- **Technical Risks:**
    - **Integration Complexity with CI/CD:** Codacity's integration with various CI/CD tools (such as GitHub, GitLab, Bitbucket, and Jenkins) could be challenging, especially with frequent updates to these platforms, potentially causing compatibility issues.
    - **Algorithm Inaccuracy and ML Model Adaptability:** Codacity's AI-powered code analysis may face accuracy challenges, particularly when adapting to different coding standards or unique project structures.
    - **Data Security and Compliance Risks:** Since Codacity handles sensitive code from sectors like finance and healthcare, there's a significant risk related to data privacy and compliance (e.g., HIPAA), especially if data breaches or unauthorized access occur.
    - **Scalability Issues with Increased Demand:** As Codacity's adoption grows, the platform may struggle to scale effectively to handle large codebases or simultaneous requests, which could impact performance and user experience.
- **Operational Risks:**
    - **Resource Constraints and Delays:** Limited technical resources or unexpected complexities in developing machine learning (ML) models could delay project milestones.
    - **Third-Party Dependency Risks:** Codacity's reliance on third-party APIs (e.g., CI/CD tools) introduces risks of system downtime or functionality changes if these services experience disruptions or modifications.
- **Financial Risks:**
    - **Economic Stability:** In events of recession, companies will lower the budget for software tools and expenses and overall profitability can be compromised. Codacity may delay or cancel plans to adopt new tools or technology.
    - **Budget Overruns:** If additional funds are needed for scaling, testing, or enhancing AI accuracy, there's a risk of exceeding the project's budget, especially during the scaling phase.
    - **Revenue Dependency and Market Adoption:** Delays in development or user adoption could impact revenue generation, particularly if users prefer more established competitors like SonarQube.
- **Market Risks:**
    - **Increased Competition in Automated Code Review Solutions:** With established players like SonarQube, Codacy, and DeepCode (Snyk Code) in the market, Codacity faces the risk of lagging if it fails to differentiate itself effectively.
    - **User Resistance to Automation:** Developers might be hesitant to adopt Codacity over traditional review practices due to concerns over automation reliability, especially if they perceive automated reviews as less thorough than manual reviews.

- **Environmental Risks:**
    - **Regulatory Compliance Updates:** Regulatory changes related to data protection and security, such as updates in GDPR, could impose additional operational constraints, requiring Codacity to adapt its data handling protocols.

Categorization of Risks:

- **Technical Risks:** Critical for Codacity, as they directly impact system functionality, accuracy, and reliability.
- **Operational Risks:** Related to Codacity's internal resource allocation and reliance on external systems, impacting project timelines.
- **Financial Risks:** These affect Codacity's budget management, financial sustainability and economic stability, especially in later project stages.
- **Market Risks:** External competition and user adoption rates, crucial for Codacity's market positioning and brand success.
- **Environmental Risks:** Legal and compliance-related risks that may arise from handling user data and regulatory changes.

Risk Impact Analysis

Assessment of the Potential Impact of Each Identified Risk on the Project:

- **Integration Complexity with CI/CD:** High impact, moderate likelihood. Delays in integration could affect user adoption and functionality.
- **Algorithm Inaccuracy and ML Model Adaptability:** Very high impact, high likelihood. Inaccurate analysis directly impacts user trust and satisfaction.
- **Data Security and Compliance Risks:** Very high impact, moderate likelihood. Security risks could lead to serious legal consequences and brand damage.
- **Scalability Issues with Increased Demand:** High impact, moderate likelihood. Scalability bottlenecks could limit Codacity's ability to serve high-demand users effectively.
- **Resource Constraints and Delays:** Moderate impact, high likelihood. Delays due to resource limitations may slow down development.
- **Third-Party Dependency Risks:** High impact, moderate likelihood. Dependency on third-party APIs could disrupt Codacity's functionality if external services fail.
- **Economic Stability:** High impact, high likelihood. Lower revenue, decrease POR and hindering the ability to sustain development
- **Budget Overruns:** High impact, moderate likelihood. Budget issues could necessitate cutbacks on important features.
- **Revenue Dependency and Market Adoption:** High impact, moderate likelihood. Delays in market adoption could impact Codacity's financial viability.
- **Increased Competition in Automated Code Review Solutions:** High impact, high likelihood. Without differentiation, Codacity might lose users to established competitors.
- **User Resistance to Automation:** Moderate impact, high likelihood. User hesitancy could affect early adoption rates, limiting growth.
- **Regulatory Compliance Updates:** Moderate impact, moderate likelihood. Non-compliance could result in legal issues and potential fines.

Prioritization of Risks Based on Severity and Likelihood:

A probability × impact matrix is used for prioritization:

1. **Data Security and Compliance Risks** - Very high impact, moderate likelihood.
2. **Algorithm Inaccuracy and ML Model Adaptability** - Very high impact, high likelihood.
3. **Integration Complexity with CI/CD** - High impact, moderate likelihood.
4. **Increased Competition in Automated Code Review Solutions** - High impact, high likelihood.
5. **Budget Overruns** - High impact, moderate likelihood.

Risk Mitigation Strategies

Development of Strategies to Mitigate or Minimize the Impact of Identified Risks:

- **Data Security and Compliance Risks:** Implement robust encryption protocols and compliance checks, with regular audits. Codacity can also create user guidelines for secure usage and establish partnerships with legal experts to stay compliant with evolving regulations.
- **Algorithm Inaccuracy and ML Model Adaptability:** Continuously train the ML models with real-world data and feedback from users to improve accuracy. Partner with experienced AI specialists to regularly update and fine-tune algorithms.
- **Integration Complexity with CI/CD:** Design Codacity with a modular architecture that facilitates integration with various CI/CD tools, performing regular compatibility testing with platforms like GitHub and Jenkins to avoid disruptions.
- **Increased Competition in Automated Code Review Solutions:** Regularly analyze market trends and user feedback to incorporate competitive features. Highlight unique selling points such as customizable review criteria and real-time feedback.
- **Economic Stability:** The company will also be able to retain its customers while going through the challenges of the economy with flexible pricing models, such as subscription plans with deferred payment alternatives or discounts.
- **Budget Overruns:** Monitor budget allocations closely and establish a contingency fund to address unforeseen expenses. Regular financial reviews will help in identifying any budget variances early on.

Contingency Plans for Addressing Unforeseen Challenges:

- **Resource Reallocation:** If additional technical resources are needed, Codacity can consider hiring contractors or outsourcing some work to meet deadlines.
- **Timeline Adjustments:** Adopt Agile methodology to manage project tasks in sprints, allowing flexibility in adjusting timelines as needed.
- **Infrastructure Scaling:** Develop Codacity on a cloud-based infrastructure to enable on-demand scaling based on user growth and activity levels.
- **Security Response Plan:** In case of a data breach, implement an incident response plan, including notification protocols, quick mitigation actions, and collaboration with cybersecurity experts.

Challenging Component: Alternative Strategies for Top Three Risks

Risk	Primary Strategy	Backup Strategy	Contingency Plan
Data Security & Compliance	**Implement Advanced Security Protocols** - AES-256 encryption for data at rest and TLS for data in transit. - Regular compliance audits for GDPR, HIPAA.	**Data Anonymization & Minimal Storage** - Anonymize user data and store only essential information. - Periodically purge old data for minimized exposure.	**Incident Response & Recovery Plan** - Rapid user notification and isolation of affected systems. - Engage cybersecurity experts for immediate breach response and conduct post-incident analysis.
Algorithm Inaccuracy	**Continuous Model Training & User Feedback** - Regularly update ML models with diverse data and feedback. - Assess accuracy and adjust as needed.	**Hybrid Review Model (AI + Manual)** - Combine AI-driven analysis with optional manual verification, especially for sensitive codebases.	**Algorithm Override & Customization** - Allow users to adjust or disable certain checks, giving control over algorithm reliance and minimizing workflow disruptions due to inaccuracies.
Integration Complexity (CI/CD)	**Modular Architecture & Compatibility Testing** - Adopt modular design for ease of integration. - Conduct extensive tests to ensure multi-platform functionality.	**Open API & Plugin-Based Architecture** - Offer an open API and custom plugins for tailored integration, letting users adapt Codacity to their CI/CD needs.	**Dedicated Integration Support Team** - Create a team specializing in integration support to quickly resolve compatibility issues and document common problems for future improvements.

7. Budgeting:

The budget for the software development lifecycle (SDLC) is divided into key categories, each accounting for crucial aspects of the project.

Cost Categories

Below is a detailed breakdown of these categories:

1. Requirements and Planning Analysis (10-15%)

1. **Activities**: Requirement gathering through Stakeholder interviews, feasibility studies, and risk identification and assessment.
2. **Justification**: This phase ensures that all functional, reliability,security, and performance requirements are well-defined which has complex elements like real-time feedback, CI/CD integration, and AI-driven review capabilities for AutoCodacity.
3. **Subcategories**:
    1. Requirement gathering for automated code analysis and ML integration.
    2. Feasibility study for CI/CD and multi-language support.

Subcategory	Description	Percentage	Cost
Requirements Gathering	Engaging with stakeholders to capture both functional and non-functional requirements	5-7%	$25,000-$35,000
Feasibility Studies	Conducting technical, operational, and financial feasibility studies	3-4%	$15,000-$20,000
Risk Management	Identifying risks and creating a mitigation plan	2-3%	$10,000-$15,000

2.System Design (15-20%)

1. **Activities**: Technical architecture design, UI/UX design, and technology select.
2. **Justification**: This phase ensures the system is well designed, scalable, secure, and meets both functional and non-functional requirements.
3. **Subcategories**:
    1. System architecture (database models, APIs).
    2. UI/UX design (mockups).
    3. Research and selection Technology stack.

Subcategory	Description	Percentage	Cost
Backend Architecture	Designing API, server-side logic and database.	7-10%	$35,000-$50,000
Frontend UI/UX Design	Creating mockups for the user interface	3-4%	$15,000 -$20,000
Technology Stack Selection	Researching and choosing appropriate technologies to use for implementation (programming languages, frameworks, tools)	3-4%	$15,000 -$20,000

3.Development (25-35%)

1. **Activities**: Backend, frontend development, system integration, and unit testing during development.
2. **Justification**: This phase involves the implementation of the system. A major slice of the budget is allocated to this phase for skilled labor and technical tools.
3. **Subcategories**:
    1. Backend Development: Implementing Server-side, database, API logic.
    2. Frontend Development: creation of User interfaces for interaction.
    3. ML Development: creating ML learning models for automated reviews.
    4. Integration: integrating third-party systems, external APIs, CI/CD tools.

Subcategory	Description	Percentage	Cost
Backend Development	Development of server-side code, database management, and API creation	12-15%	$60,000 - $75,000
Frontend Development	Building the user interface and integrating it with the backend system	5-7%	$25,000 - $35,000
AI/ML Development	Creating machine learning models for automated code reviews	5-8%	$25,000 - $40,000
Integration	Integrating AutoCodacity with CI/CD tools and multi-language support	3-5%	$15,000-$25,000

4.Testing & Quality Assurance (10-15%)

1. **Activities**: White Box, security testing, automated, and performance testing.
2. **Justification**: This phase ensures software is not only bug-free but also secure by investing in testing that prevents expensive post-release fixes.
3. **Subcategories**:
    1. Functional testing (testing of features).
    2. Automated testing (tools for repetitive test cases).
    3. Security testing (vulnerability assessments).
    4. Performance testing (load testing)

Subcategory	Description	Percentage	Cost
Functional Testing	Testing at functions level, including code reviews, refactoring, and real-time feedback	5-7%	$25,000-$35,000
Security Testing	Ensuring system is tested against all vulnerabilities	2-3%	$10,000-$15,000
Performance Testing	System performance is tested against heavy loads	2-3%	$10,000-$15,000
Automated Testing Tools	Usage of automated testing tools to streamline QA processes	1-2%	$5,000 - $10,000

5.**Deployment (5-10%)**

1. **Activities**:Configuring production environment, documentation, and user training.
2. **Justification**: This phase ensures software is deployed successfully and operates as expected in the production environment.
3. **Subcategories**:
    1. Production environment configuration.
    2. Technical and user documentation.
    3. Training and support for end-users.

Subcategory	Description	Percentage	Cost
Production Environment	Configuring environment especially cloud infrastructure for scalability to deploy software for market.	3-5%	$15,000 - $25,000
Documentation	Preparing technical and user documentation for deployment and usage	2-3%	$10,000 - $15,000

Training	Providing sessions to train developers and end-users	1-2%	$5,000 - $10,000

6.**Marketing (10-15%)**

1. **Activities**: Market analysis, branding, digital marketing campaigns, and sales.
2. **Justification**: This phase attracts users by promoting software for use.
3. **Subcategories**:
    1. Market analysis.
    2. Digital marketing (advertisements on social media).
    3. Content creation (blogging, case studies).
    4. Sales teams and customer outreach.

Subcategory	Description	Percentage	Cost
Digital Marketing	Online campaigns, and social media outreach	4-6%	$20,000-$30,000
Content Marketing	Blogs, case studies, and other educational content targeting developers	3-4%	$15,000-$20,000
Branding and Design	Creating branding/ visual identity for AutoCodacity (logo).	2-3%	$10,000-$15,000
Sales Outreach	Direct outreach to enterprises and market.	1-2%	$5,000 - $10,000

7. Ongoing Maintenance (15-25%)

1. **Activities**: security patches, bug fixes, updates, new feature development.
2. **Justification**: This phase ensures software is functional, secure, and reliable.
3. **Subcategories**:
    1. Bug fixing.
    2. Security patching.
    3. Feature enhancement and updates.
    4. Performance optimization.

Subcategory	Description	Percentage	Cost
Bug Fixes	Identifying and resolving bugs post-deployment	5-7%	$25,000 - $35,000

Security Patches	Regular updates to ensure security compliance	3-4%	$15,000 - $20,000
Feature Updates	Updating or adding new features to ensure the effectiveness of system	5-7%	$25,000 - $35,000
Performance Optimization	Ensuring system performance scales with increased demand	2-3%	$10,000 - $15,000

8. Customer Support (5-10% of Total Budget)

Activities: technical support, handling inquiries, and managing feedback from developers.
Justification: This ensures AutoCodacity users have ongoing assistance after deployment.

Subcategory	Description	Percentage	Cost
Technical Support	Providing ongoing technical support for resolving user issues	3-5%	$15,000 - $25,000
User Feedback Management	Collecting and addressing user feedback for potential feature enhancements	1-2%	$5,000 - $10,000
Training & Documentation Updates	Continuous updates to training materials and user guides based on evolving features	1-2%	$5,000 - $10,000

Allocation of Funds to Each Category

The allocation of funds is based on the complexity, quality standards, and scope of the project as **development** and **maintenance** are allocated higher percentages due to their critical role in the project's success and long-term sustainability. This allocation is informed by quality practices and the nature of the project as complex integrations require a higher budget for ongoing maintenance and system updates.

Summary of Budget Allocation for AutoCodacity

Category	Percentage of Total Budget	Estimated Cost (for $500,000 project)
Planning & Requirements	10-15%	$50,000 - $75,000
System Design	15-20%	$75,000 - $100,000

Development	25-35%	$125,000 - $175,000
Testing & QA	10-15%	$50,000 - $75,000
Deployment	5-10%	$25,000 - $50,000
Marketing	10-15%	$50,000 - $75,000
Ongoing Maintenance	15-25%	$75,000 - $125,000
Customer Support	5-10%	$25,000 - $50,000

Resource Costing

Effort Estimation Models for AutoCodacity

1. COCOMO (Constructive Cost Model) : is a model that estimates the effort (in person-months), cost, and schedule of software projects based on the project size and the project's complexity.

COCOMO Model Steps:

1. **Estimate Project Size (KLOC)**:
    - AutoCodacity, estimate the number of lines of code for each major component (backend, frontend, AI/ML models).
    - Example: Backend (10,000 LOC), Frontend (5,000 LOC), AI/ML models (3,000 LOC), giving a total of **43 KLOC**.
2. **Determine the Project Type**: AutoCodacity would likely fall under the **Semi-Detached** category due to its AI, security, and multi-language support.
3. **Use the COCOMO Formula**:
   The basic COCOMO formula for effort is:

   Effort (person-months) = a× (KLOC) b

   Where:

    - **a** and **b** are constants based on the project type.
    - For a semi-detached project, **a = 3.0** and **b = 1.12**.

   Applying the formula for **AutoCodacity**:

   Effort=3.0× (43)1.12= 202.5.

   This means that developing AutoCodacity will require about **67.86 person-months** of effort.

   COCOMO Output:

- **Effort**: 202.5 person-months (approximately).
- **Team Size**: If you assign 15 developers to the project, the estimated timeline would be around **13.5 months** (202.5 person-months ÷ 15 developers).

2. Function Point Analysis (FPA)

Function Point Analysis (FPA): used as another effort estimation technique. It focuses on **functionality** rather than the lines of code, Useful when the scope of the project is well-defined in terms of inputs, outputs, and user interactions.

FPA Calculation Steps**:**

1. **Identify Functions**: Break the project down into five major components:
    - **Inputs**: Data entered by users (code submitted for review).
    - **Outputs**: Results generated by the system (real-time code feedback).
    - **Inquiries**: Requests for data developers requesting a summary of code review results).
    - **Internal Logical Files (ILF)**: Data stored by the system (user profiles, code review history).
    - **External Interface Files (EIF)**: Data accessed from external systems (CI/CD tools like GitHub).
2. **Assign Weights to Each Function**: Each function type is weighted based on its complexity. Range would be from 3 to 7.
3. **Calculate Function Points (FP)**: Sum the weighted individual component values to calculate the total number of function points.

Calculation for AutoCodacity:

- **Inputs** (code submission): 42 inputs × 4 (simple)= 168FP.
- **Outputs** (review reports): 40 outputs × 5(complex) = 200FP.
- **Inquiries** (summary requests): 48 inquiries × 4(simple) = 192FP.
- **ILFs** (user profile storage): 8 ILFs × 10 (average) = 80FP.
- **EIFs** (data from CI/CD): 4 external interfaces × 5(simple) = 20FP.

Unadjusted Function Point=168+200+192+80+20=660 function points.

Value Adjustment Factor= 52*0.01+0.65=1.17

Adjusted Function Point:772.2

4. **Determine Effort per Function Point**: Using industry standards, the typical effort required per function point is around 6 to 8 hours.

Effort= 772.2 FP×8 hours/FP=6177.6 hours

FPA Output:

- **Total Effort**: 6177.6 hours (for the entire team).

Comparison of Models for AutoCodacity:

Model	Effort Estimation	Time Estimation	Team Size
COCOMO	202.5 person-months	13.5 months	15 developers
FPA	6177.6 hours (total)	2.57 months	15 developers

COCOMO provides a more robust estimate for AutoCodacity because it considers the overall size (KLOC) and complexity (semi-detached project).

FPA is useful when focusing on the functionality provided to the user, but it tends to estimate lower effort for projects involving complex AI and ML components.

Breakdown of Resources and Costs

Human Resources
The following table breaks down the cost of human resources based on their hourly rates and the total effort required.

Resource	Hourly Rate	Person-Months	Total Cost
Backend Developer	$90/hour	5 person-months	$90/hour × 160 hours/month × 5 PM = $72,000
Frontend Developer	$90/hour	4 person-months	$90/hour × 160 hours/month × 4 PM = $57,600
AI/ML Engineer	$100/hour	3 person-months	$100/hour × 160 hours/month × 3 PM = $48,000
QA Tester	$75/hour	2 person-months	$75/hour × 160 hours/month × 2 PM = $24,000
Project Manager	$110/hour	1 person-months	$110/hour × 160 hours/month × 1 PM = $17,600

Rationale for how effort is distributed across roles:

160 hours/month is used as a standard for full-time work, assuming 40 hours per week over 4 weeks.

Role	Person-Months	Reason for Allocation
Backend Developer	5 person-months	Backend development is crucial for AI/ML integration, database setup, and secure handling of large-scale code reviews. Backend work is more complex, requiring more effort.
Frontend Developer	4 person-months	The user interface (UI) will be equally complex compared to backend tasks, to ensure real-time feedback and integration with the backend.
AI/ML Engineer	3 person-months	AI and machine learning are key components of AutoCodacity, so significant effort is required to develop, train, and fine-tune models for automated code review.
QA Tester	2 person-months	Quality assurance, functional testing, security testing, and performance testing are critical to ensure functionality is working as expected to avoid post deployment risks and cost.
Project Manager	1 person-months	Project management is needed throughout the entire lifecycle to ensure timelines, team coordination, and stakeholder communication, especially in a complex AI-driven project.

Technology and tools include licenses for various software, cloud services for hosting, and the cost of machine learning and AI frameworks.

1. Cloud Hosting (AWS, GCP, or Azure)

Rationale: AutoCodacity requires cloud infrastructure for hosting backend, AI/ML models, and running CI/CD pipelines. The system will handle significant workloads as it processes code reviews and security checks, requiring scalable cloud solutions.

Cost Derivation:

- **Storage**: Estimated 1 TB of storage for large codebases, ML datasets, and logs at $0.023/GB/month (AWS S3 pricing).
- **Compute Instances**: 3 EC2 compute instances for backend, AI/ML processing, and integration testing at $100/month each.
- **Data Transfer**: Estimating 10 TB of data transferred in and out per month at $0.09/GB.
- **Total Cloud Hosting Cost**: $68,800

2. AI/ML Tools (TensorFlow, PyTorch)

Rationale: AutoCodacity's automated code review system works heavily on machine learning for code analysis and refactoring models. TensorFlow and PyTorch, being open-source frameworks, have free tiers but may require additional paid support and services for production-level deployment.

Cost Derivation:

- **TensorFlow Enterprise Support**: Production support for AI/ML models is critical for maintaining model performance.
- **GPU/TPU Compute for AI Training**: Depending on the complexity of the ML models, renting GPUs/TPUs can cost $0.50 to $3 per hour. Assuming an average of **100 hours/month** of GPU usage at **$1/hour**:
- **Total AI/ML Tool Costs**: Support Cost + GPU Usage=$27,500/year

3. **CI/CD Tools (Jenkins, GitHub, GitLab)**

Rationale: AutoCodacity requires seamless integration with CI/CD pipelines for automated code reviews. Jenkins (for pipeline management) and GitHub (for repository management) are best fit.

Cost Derivation:

- **Jenkins**: Jenkins is open-source, but hosting it on cloud infrastructure incurs compute and storage costs. These costs are bundled with cloud hosting estimates.
- **GitHub Enterprise**: GitHub offers advanced collaboration features and is necessary for managing repositories with multiple users.
- **Total CI/CD Tool Costs**: $35,500/year

4.Testing Tools (Selenium, JIRA, Postman)

Rationale: Automated testing tools such as Selenium (for UI testing) and JIRA (for issue tracking) are required for QA efforts. These tools help identify bugs, test performance, and manage bug-fixing processes.

Cost Derivation:

- **Selenium (Open Source)**: No direct licensing fees, but requires integration and server costs for execution, which are accounted for in cloud hosting.
- **JIRA**: JIRA is required for issue tracking and agile project management.
- **Postman Pro**: API testing and monitoring tool used to ensure backend services functionality.
  Total Testing Tool Costs: JIRA + Postman Pro = $15,000/year

Tool/Service	Estimated Cost	Justification
Cloud Hosting (AWS)	$68,800/year	Estimated based on typical usage for CI/CD and storage.
AI/ML Tools (TensorFlow, PyTorch)	$27,500/year	Estimated cost for using open-source AI/ML frameworks with paid support.
CI/CD Tools (Jenkins, GitHub)	$35,500/year	Includes licenses for integration with AutoCodacity's CI/CD pipeline.
Testing Tools (Selenium, JIRA)	$15,000/year	Tools for automated and manual testing, including bug tracking.

External Services
1.Security Consultant (External Audit for Security Compliance)

Rationale: AutoCodacity can make security compliance critical. An external security consultant will perform audit code for vulnerabilities, and ensure compliance with industry regulations like GDPR.

Cost Derivation:

- **Consultant Rate**: An hourly rate for a security consultant is **$110/hour**.
- **Time Required**: Security audits typically take **160 hours** for a project of this scale, including system architecture reviews, code audits, and compliance checks.

2.Marketing Agency (Digital Marketing and Branding)

Rationale: A marketing agency will manage AutoCodacity's branding, digital campaigns, and customer acquisition. For AutoCodacity's target audience (developers, startups, large enterprises), a comprehensive marketing campaign is required.

Cost Derivation:

- **Agency Fees**: Digital marketing agencies usually charge around **$3280/month** for a full-service package, including content creation, paid advertisements, and social media management.
- However, based on AutoCodacity's initial needs, a smaller campaign might require about 5 months of intense marketing, reducing the total cost.

Service	**Cost**	**Justification**
Security Consultant	$17,600 (one-time)	External security audit to ensure compliance with industry regulations like GDPR.
Marketing Agency	$16400 (5 months)	Digital marketing, content creation, and branding for targeting developers and enterprises.

Contingency Budget

Allocation of a Contingency Budget : A contingency budget of **10%** of the total budget is recommended to account for unforeseen expenses, such as:

- **Unexpected delays** in development or testing.
- **Scope changes** requested by stakeholders.
- **Additional resources** required for handling more complex bugs or security issues.

Contingency Budge	**Percentage of Total Budget**	**Estimated Cost**
Contingency	20%	$100,000

Explanation behind the Contingency Budget

The **10% contingency budget** is justified based on several key risks and high-impact activities within the AutoCodacity project. These risks affect costs, resource allocation, and timelines.

1. AI/ML Integration Risks
AutoCodacity's AI and machine learning components introduce technical risks associated with developing, training, and fine-tuning AI models include:

- **Model Accuracy and Performance**: AI models may require additional iterations to reach the desired level of accuracy for automated code reviews. This could lead to **increased development effort** for AI/ML engineers.
- **Unexpected Algorithmic Complexity**: As AutoCodacity's machine learning system learns from code reviews, unforeseen **complexity in the algorithms** could require more computational resources or specialized skills to optimize the performance.

Potential Impact: Additional time for AI/ML model tuning and testing, leading to higher resource costs for AI/ML engineers and extended development time.

2. CI/CD and Multi-Language Support Integration Risks
AutoCodacity's system needs to integrate seamlessly with existing CI/CD pipelines (e.g., Jenkins, GitHub) and support multiple programming languages. The integration process can encounter several challenges:

- **Integration Failures**: Unexpected issues while integrating CI/CD tools could cause delays and require additional developer hours to resolve compatibility problems.
- **Extending Multi-Language Support**: Extension to a wide range of programming languages (e.g., Python, Java) can introduce complexities that might require additional time to develop custom solutions for language-specific issues.

Potential Impact: Extended development time for backend developers, potentially higher cloud infrastructure costs for testing multiple CI/CD pipelines and language environments.

3. Security Compliance Risks

Given the importance of security for AutoCodacity, meeting compliance standards like GDPR can lead to increased costs:

- **Security Vulnerabilities**: If vulnerabilities are discovered during testing or after deployment, immediate remediation is required, which can delay project completion.
- **Additional Security Audits**: If stakeholders require more in-depth security audits, this may require hiring external consultants or security specialists.

  Potential Impact: Increased resource costs for security audits, compliance validation, and potential security consultant fees.

4. Scope Creep and Stakeholder Changes

As with most software development projects, **scope creep** is a common risk.

- **Additional Features**: Stakeholders may request new features or enhancements during the development process, which could require extra resources, increased development time, and additional testing cycles.
- **User Feedback-Driven Changes**: Early feedback from developers or users might require modifications or additional backend features to improve the system's functionality.

Potential Impact: Increased costs for backend and frontend developers, as well as additional QA and testing efforts.

5. Delays in Testing and Debugging

AutoCodacity's automated code review system is expected to handle real-time code analysis across multiple environments and programming languages. The testing and debugging process for such a system is inherently complex:

- **Performance Bottlenecks**: During testing, performance bottlenecks may be discovered that require optimization, especially when the system is tested under load.
- **Increased QA Time**: Functional, performance, and security testing across multiple language environments could take longer than anticipated, extending the testing phase.

Potential Impact: Additional costs related to QA testers, cloud infrastructure for testing, and possibly extending the project timeline.

Below is a breakdown of how contingency funds may be distributed across different phases:

Project Phase/Activity	Potential Risk	Allocation of Contingency Budget
AI/ML Development & Testing	Complexity in training models, algorithmic issues	30% ($30,000)
CI/CD & Multi-Language Integration	Unexpected integration failures, language-specific complexities	20% ($20,000)
Security Compliance & Audits	Security vulnerabilities, compliance requirements	20% ($20,000)
Scope Creep	New feature requests from stakeholders, additional functionalities	20% ($20,000)
Extended Testing & Debugging	Increased QA time, debugging performance issues	10% ($10,000)

Total Cost Breakdown

Calculating the total cost for AutoCodacity based on all the resources involved:

Category	Allocated Budget	Percentage of Total Budget
Human Resources	$219,200	43.84%
Technology & Tools	$146,800	29.36%
External Services	$34,000	6.8%
Contingency Plan	$100,000	20%
Total	**$500,000**	100%