ResolveIT Helpdesk WebApp
Comprehensive SDLC Documentation
Enterprise Application Development
IDS517

https://github.com/t4n15hq/IDS517-Helpdesk-Webapp
https://resolveituic.tech/

Group 5 :
Akshay Shenoy
Ashna Sheregar
Kamal Teckchandani
Nonitha Vampati
Omkar Nehete
Saatvik Shukla
Tanishq Padwal

ResolveIT: Streamlined Technology Support System

ResolveIT Development Team:

Tanishq Padwal:

- **Frontend Development:** Implementing the user interface with attention to responsiveness and interactivity.
- **Backend Integration:** Ensuring seamless data flow between the frontend and backend services.
- **API Development:** Designing and implementing RESTful APIs for client-server communication.
- **Hosting:** Deploying the application on a suitable platform and managing domain configuration.
- **Project Planning:** Overseeing the project timeline, task allocation, and meeting milestones.

Nonitha Vampati:

- **Google Cloud SQL Integration:** Setting up and configuring the database services in Google Cloud.
- **Cloud-Related Activities:** Managing cloud resources, including storage, computing, and networking in Google Cloud.
- **Hosting:** Deploying the application on a suitable platform and managing domain configuration.

Kamal Teckchandani:

- **Login Integration:** Implementing secure authentication mechanisms.
- **Mail Integration:** Setting up automated email notifications for various user actions.
- **Database Schema:** Designing the database structure for efficiency and scalability.
- **Cloud-Related Activities:** Assisting with cloud configurations and optimizations.

Akshay Shenoy:

- **Admin User Panel Development:** Building the admin interface for user management and analytics.
- **Email Notification System:** Enhancing the mail system to include customizable templates and triggers.
- **Suggesting Improvements:** Analyze the current system and propose improvements based on feedback and technological advancements.

Ashna Sheregar:

- **UI/UX Design:** Creating intuitive and appealing design concepts for the application.
- **Testing:** Conducting thorough testing to identify and fix issues.
- **Documentation:** Writing comprehensive documentation for the system architecture and user guides.
- **Debugging and User Acceptance Testing (UAT):** Leading efforts in debugging and conducting UAT to gather feedback.
- **Bug Tracking and Resolution:** Manage the bug tracking system and resolve issues promptly.

Omkar Nehete and Saatvik Shukla:

- **Service Request Management:** Developing the system for handling and tracking service requests.
- **User Manual and Technical Documentation:** Creating detailed guides for users and developers.
- **Database Schema Design:** Collaboratively working on the database design to ensure it meets all application needs.
- **Milestone Tracking:** Keeping track of project milestones and reporting on progress.
- **Suggesting Improvements:** Analyze the current system and propose improvements based on feedback and technological advancements.

Team Meets Schedule :

Daily Stand-up (Monday, Wednesday & Friday)
- Time: 10:00 PM (15 minutes)
- Purpose: Quick check-in to discuss the day's accomplishments and plans for the next and highlight any blockers.

Communication and Progress Updates (Tuesday and Thursday)
- Time: 10:30 PM (30 minutes)
- Purpose: Share individual progress, address challenges encountered, and collaborate on solutions.

Development Sprints for Helpdesk System (Monday, Wednesday, and Friday)
- Dedicated Time for Development: Post-daily stand-up till 12:00 AM
- Purpose: Focus on development tasks such as building new features, debugging, and making enhancements. Team members collaborate and are available for consultations.

Testing and Quality Assurance (Tuesday and Thursday)
- Time: Post-progress update meeting till 12:00 AM
- Purpose: Engage in targeted testing phases, including unit tests, integration tests, and manual exploratory testing, to ensure the application's quality.

Feature Brainstorming and Planning (Wednesday)
- Time: 10:30 PM (1 hour)
- Purpose: Ideate on future features, enhancements, and process improvements. This session also serves to prioritize upcoming tasks and features.

Integration Testing and Bug Reporting Session (Friday)
- Time: 10:30 PM - 12:00 AM
- Purpose: Conduct thorough integration testing to verify the interoperability of different system components. Log in and discuss any discovered bugs or compatibility issues.

Code Maintenance and Review (Thursday and Saturday)

- Time: Thursday 10:30 PM - 11:30 PM, Saturday (Flexible, as required)
- Purpose: Dedicated time for code refactoring, optimizing existing implementations, addressing technical debt, and conducting peer code reviews to maintain code quality.

Team Progress :

Advanced Database Operations:

- Progress: The web application now supports real-time data updates and a robust comment system for tracking service requests. This interactive feature enhances user experience by allowing dynamic updates and discussions within the browser, moving beyond the static nature of the Eclipse-based system.

Supervisor Review Process:

- Progress: The web application has introduced a comprehensive supervisor review and priority adjustment workflow. This new system enables supervisors to efficiently review, prioritize, and manage service requests through a streamlined web interface, significantly improving the desktop application's capabilities.

Automated Email Notifications:

- Progress: The web application has been equipped with an automated email notification system. This feature ensures timely updates on service request statuses are sent directly to users and staff, facilitating improved communication and engagement without dependency on desktop-based email solutions.

Security Enhancements:

- Progress: Enhanced security measures have been implemented, including password hashing and secure database connections. These improvements fortify the web application against common security threats and vulnerabilities, providing a safe environment for managing sensitive user data, a crucial advancement from the previous IDE-based security approach.

Client-Server Architecture:

- Progress: A scalable client-server architecture has been successfully established, leveraging modern web technologies for frontend and backend development.

Business Area :

The Information Technology (IT) Support Services sector is the selected business area for implementing the Online Helpdesk System. This sphere is integral to the operational backbone of any modern enterprise, acting as the first line of defense against technical disruptions that can impede business processes. It encompasses a broad range of services, including but not limited to hardware maintenance, software troubleshooting, network management, and cybersecurity defenses.

Technology underpins virtually every aspect of business operations; the role of IT Support Services extends beyond mere maintenance. It is about enabling and empowering the workforce through technology, ensuring that all systems function seamlessly, and providing swift resolutions to technical issues that could otherwise lead to productivity loss.

The Online Helpdesk System is the hub for all IT support activities, offering a central platform for employees to report issues, track their status, and receive assistance. It streamlines ticket management, prioritizes problems, and allocates resources efficiently for prompt issue resolution.

- **Ticket Submission:** Employees can report IT issues with detailed descriptions and attach relevant information directly within the web application, streamlining the troubleshooting process.

- **Ticket Tracking:** Users can view real-time status updates of their tickets from submission through to resolution, accessible via the web interface.

- **Resource Allocation:** IT support staff can efficiently assign tickets to the appropriate team members based on availability and expertise, ensuring optimal handling of each issue.

- **Resolution Workflow:** Implements a structured resolution workflow in the web application, which includes diagnosis, action steps, follow-up, and closure, enhancing systematic troubleshooting.

- **Client-Server Architecture Implementation:** Enhanced the interaction between the client-side user interface and server-side database management through a robust client-server model, allowing real-time updates and synchronization.

- **Enhanced Security Measures:** Advanced security protocols such as password hashing and secure database connections have been integrated to protect the web application's sensitive data and user information.

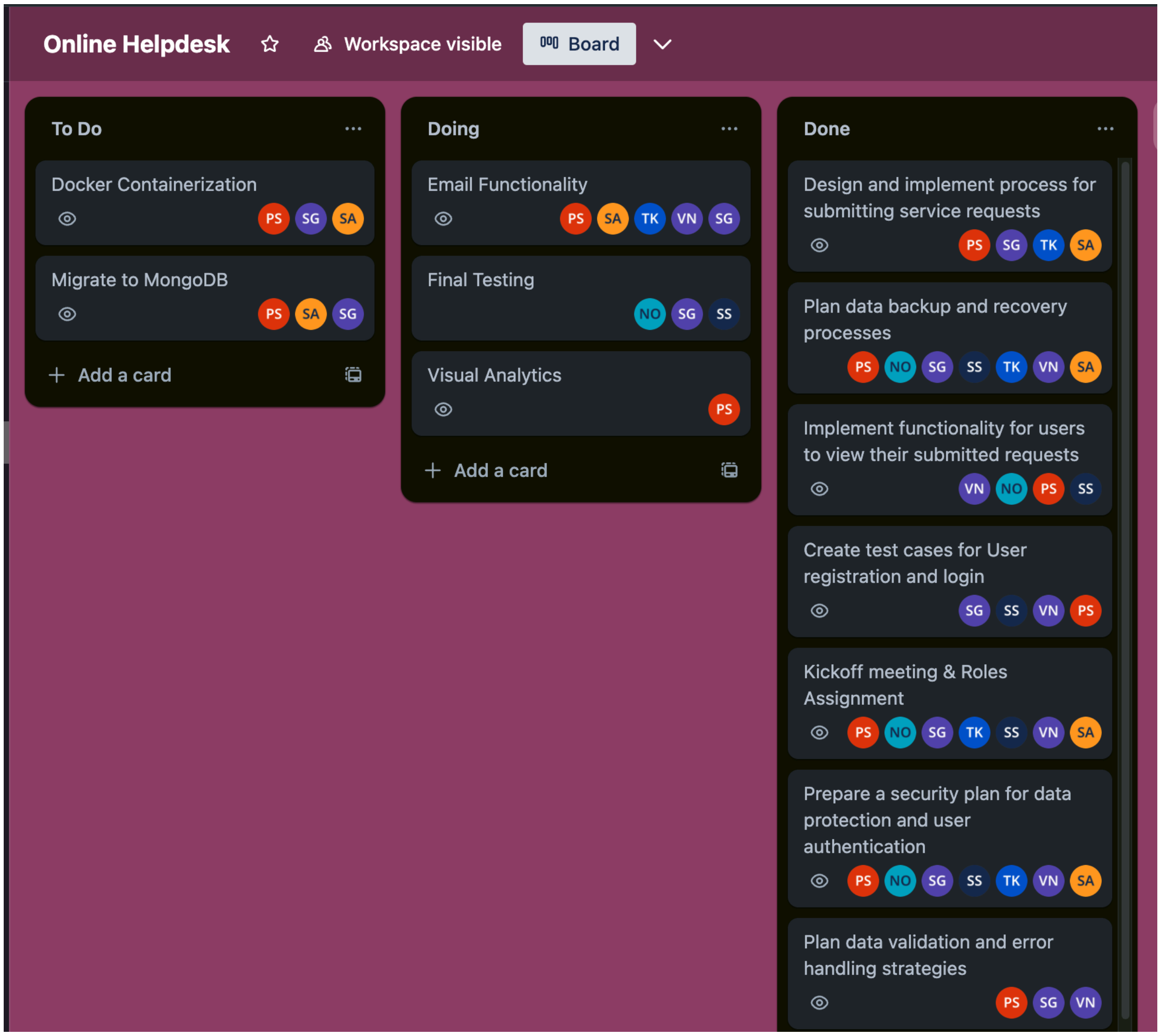
- **User Experience Optimization:** Refined the user interface design in the web application to reduce complexity and enhance user engagement, featuring intuitive navigation and interaction elements.

- **Automated Notifications:** Developed a computerized notification system within the web application to keep users and IT staff updated on ticket progress and important system notifications.

- **Expanded Support Resources:** Included a comprehensive knowledge base and FAQ section in the web application, providing users with immediate self-help options and reducing the need for ticket submissions for common issues.

- **Service Level Agreement (SLA) Tracking:** Integrated SLA tracking within the web application to monitor and ensure timely responses and resolutions, maintaining adherence to service standards.

Trello Board :


Online Helpdesk

To Do

- Docker Containerization — PS SG SA
- Migrate to MongoDB — PS SA SG

Doing

- Email Functionality — PS SA TK VN SG
- Final Testing — NO SG SS
- Visual Analytics — PS

Done

- Design and implement process for submitting service requests — PS SG TK SA
- Plan data backup and recovery processes — PS NO SG SS TK VN SA
- Implement functionality for users to view their submitted requests — VN NO PS SS
- Create test cases for User registration and login — SG SS VN PS
- Kickoff meeting & Roles Assignment — PS NO SG TK SS VN SA
- Prepare a security plan for data protection and user authentication — PS NO SG SS TK VN SA
- Plan data validation and error handling strategies — PS SG VN

Project scheduling: Task List

April 1st - April 3rd: Initial Project Planning and Environment Setup
- Tanishq Padwal oversees the transition assessment, realigning goals based on the web application framework. Sets up the project planning and scheduling.
- Nonitha Vampati begins setting up Google Cloud SQL and explores cloud-based services for future integration.
- Kamal Teckchandani assists with cloud setup and begins planning for login and mail integration features.

April 4th - April 6th: Frontend and Backend Development Kick-off
- Tanishq Padwal sets up the web development environment for the frontend (HTML/CSS/JavaScript) and backend (Node.js).
- Nonitha Vampati and Kamal Teckchandani collaborate on integrating Google Cloud SQL with the backend, focusing on real-time data updates.

April 7th - April 9th: Feature Development and Security Planning
- Akshay Shenoy started developing the admin user panel and planned the email notification system.
- Ashna Sheregar designs initial UI/UX prototypes and builds a testing framework.
- Omkar Nehete and Saatvik Shukla drafted the initial database schema and service request management features.

April 10th - April 12th: Advanced Feature Integration and Security Enhancements
- Tanishq Padwal leads the integration of API making, focusing on client-server communication.
- Nonitha Vampati continues with cloud integration, ensuring cloud services support the application's data management.
- Akshay Shenoy implements the email notification system.

April 13th - April 15th: UI/UX Implementation and Review
- Ashna Sheregar finalizes UI/UX design elements and collaborates with Tanishq for front-end development.

- Kamal Teckchandani tests and debugs the login and mail integration features.
- Omkar Nehete and Saatvik Shukla refine the service request management system based on the database schema.

April 16th - April 18th: Testing, Debugging, and Performance Optimization

- All Team Members undergo a comprehensive testing phase, including unit, integration, and performance tests.
- Tanishq Padwal and Akshay Shenoy focus on performance optimization for backend operations and the admin panel.

April 19th - April 21st: Documentation and Final Preparations

- Ashna Sheregar, Omkar Nehete, and Saatvik Shukla prepared user manuals and technical documentation and finalized the bug-tracking system.
- Kamal Teckchandani and Nonitha Vampati document all cloud functionalities and perform final security checks.

April 22nd: Project Review and Deployment Planning

- All Team Members participate in a deployment readiness review, confirming the application is fully functional, secure, and ready for launch.

Data Management :

Database Schema and Structure:
Google Cloud SQL Instance: Leveraging Google Cloud SQL ensures a reliable, scalable, and secure database service that supports the application's data storage needs. It facilitates efficient data management and real-time updates crucial for the helpdesk system.

Users Table:
- Purpose: Manages user data, supporting authentication and role-based access control.
- Fields include UserID, Name, Email, Role, and Password. The schema ensures unique identification, secure authentication, and role differentiation between employees and IT staff.

ServiceRequests Table:
- Purpose: Tracks the lifecycle of service requests from submission to resolution.
- Fields include RequestID, Problem, Severity, SubmittedBy, Description, Priority, AssignedTo, Status, Comment, Timestamp, ResolutionDate, and FilePath. This design captures comprehensive details about each service request, facilitating effective tracking and resolution.

Data Flow and Operations:
- Ticket Submission:
- Users log in and submit service requests with details such as the problem description, severity, and attachments. This initiates a record in the ServiceRequests table.
- Data validation ensures the integrity of submitted information.

Ticket Assignment:
- IT staff reviews and assigns tickets, updating the AssignedTo and Priority fields. This process relies on efficient querying and updates, supported by indexed fields for performance optimization.
- The foreign keys link was submitted and assigned to the users' table, maintaining data consistency.

Issue Resolution:

- IT staff updates the ticket's status and may add comments to document the resolution process. The Status and Comment fields facilitate this tracking.
- Upon resolution, the Status is set to "Resolved," and the ResolutionDate is recorded, marking the ticket's lifecycle end.

Notification and Closure:

- Automated notifications inform users of ticket resolutions. This process integrates with the email system, leveraging user emails from the Users table.
- Users can review and close tickets, confirming the satisfactory resolution.

Security and Performance Optimization:

- Security Enhancements: Passwords are stored hashed for security. Secure connections to Google Cloud SQL protect data in transit. Regular security assessments should identify and mitigate vulnerabilities.
- Performance Optimization: Indexes on frequently accessed fields like Email, Status, and AssignedTo improve query performance. Cloud SQL's scalability options should be utilized to handle varying loads, ensuring responsive application performance.
- Data Integrity: Enforce referential integrity through foreign keys and use constraints to ensure data validity (e.g., email format, predefined values for Severity and Priority).

Backup and Recovery:

- Regular backups and a clear recovery plan are crucial for disaster recovery. Google Cloud SQL's automated backup and point-in-time recovery features should be configured to safeguard against data loss.

Monitoring and Maintenance:

- Monitoring database performance and regular maintenance tasks, such as cleaning up old entries and optimizing queries, ensure the system remains efficient and scalable.

System Architecture Overview

The "ResolveIT" Helpdesk Web Application is architected using a contemporary client-server model, significantly enhancing its scalability, security, and ease of maintenance. This sophisticated architecture bifurcates the application into two fundamental components: the web client for engaging user interfaces and the server for robust backend operations.

Client Application:

- Technologies Employed: The front end is crafted with HTML, CSS, and JavaScript, creating an engaging and dynamic user interface (UI) compatible across web browsers.
- Key Features: The application boasts features for account creation, login procedures, submission and tracking of service requests, and comprehensive administrative functionalities. Its intuitive and responsive design facilitates a frictionless user experience on various devices.
- Communication Mechanisms: HTTP/HTTPS requests achieve interaction with the server. AJAX is employed for asynchronous data retrieval, ensuring the application remains responsive without page reloads and enhancing user experience.

Server Configuration:

- Core Framework: The server side is powered by Node.js with Express, which efficiently manages HTTP requests from the client. It oversees crucial operations such as user authentication, session management, and request routing.
- Middleware Integration: It incorporates several middleware solutions, including body-parser for parsing requests, cors for enabling Cross-Origin Resource Sharing, express-session for session management, and multer for file upload capabilities.
- Security Protocols: Security is fortified with bcrypt for hashing passwords, thus securing user authentication and protecting sensitive data.

Database Management:

- DBMS Selection: The application utilizes Google Cloud SQL for database management, supporting scalable MySQL instances. This choice provides

a reliable platform for storing and managing critical data such as user profiles, service requests, and comments.
- Schema Design: The database schema is meticulously designed for optimized data retrieval and updates, with structured tables dedicated to users, service requests, and other pertinent information.

Communication and Data Flow:
- API Endpoints: The backend exposes RESTful API endpoints consumed by the client to facilitate actions like user registration, login, service request submissions, and tracking updates.
- Data Transmission Protocol: Secure HTTPS connections ensure encrypted data exchange between client and server. JSON, known for its lightweight and adaptability, is the primary data exchange format that promotes efficient data handling.

Operational Workflow:
- The web application is designed with a clear operational workflow guiding users from the initial account registration through to the service request resolution process. This workflow includes mechanisms for ticket tracking, prioritization by severity, and facilitating direct communication between users and the IT support staff to guarantee timely and efficient issue resolution.

Process Flow

User Registration and Login:

- **Start:** The process begins when a user interacts with the system.
- **User Registration:**
    - Users select the "Register" option.
    - They fill out and submit the registration form.
    - The Helpdesk system creates a new user record.
    - Users are then redirected to the homepage.
- **User Login:**
    - Returning users choose the "Login" option.
    - They enter their credentials in the login form.
    - Upon successful validation, they are redirected to the Service Request Homepage.

Service Request Process:

- **Submit/View Request:**
    - Users can either submit a new request or view existing requests from the Service Request Homepage.
    - If submitting, they complete a service request form, which includes a section for attaching files if necessary. An email alert is sent to the helpdesk team once the request is saved.
    - If viewing, they're presented with a list of their requests.
- **Knowledge Base/FAQ:**
    - Before submitting a request, users are encouraged to consult the Knowledge Base/FAQ section for self-service solutions.
- **Supervisor Interaction:**
    - Supervisors have the option to log in from the Service Request Homepage.
    - They are redirected to a Supervisor Dashboard where they can view all service requests.
    - Supervisors have the option to update the priority of tickets as needed.

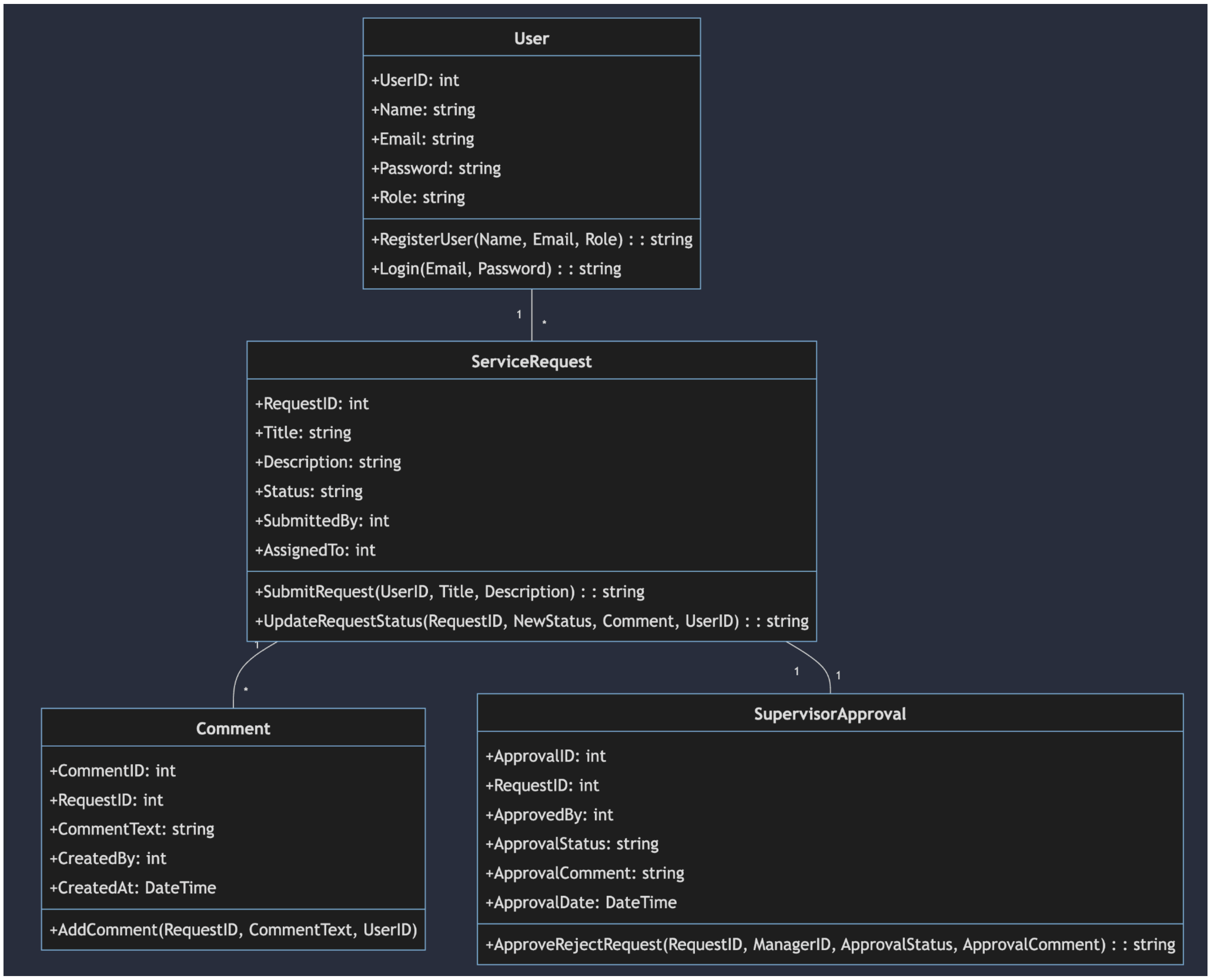
Admin Panel:

- **Admin Panel Access:**
    - If the user is an admin, they can access the Admin Panel after login.

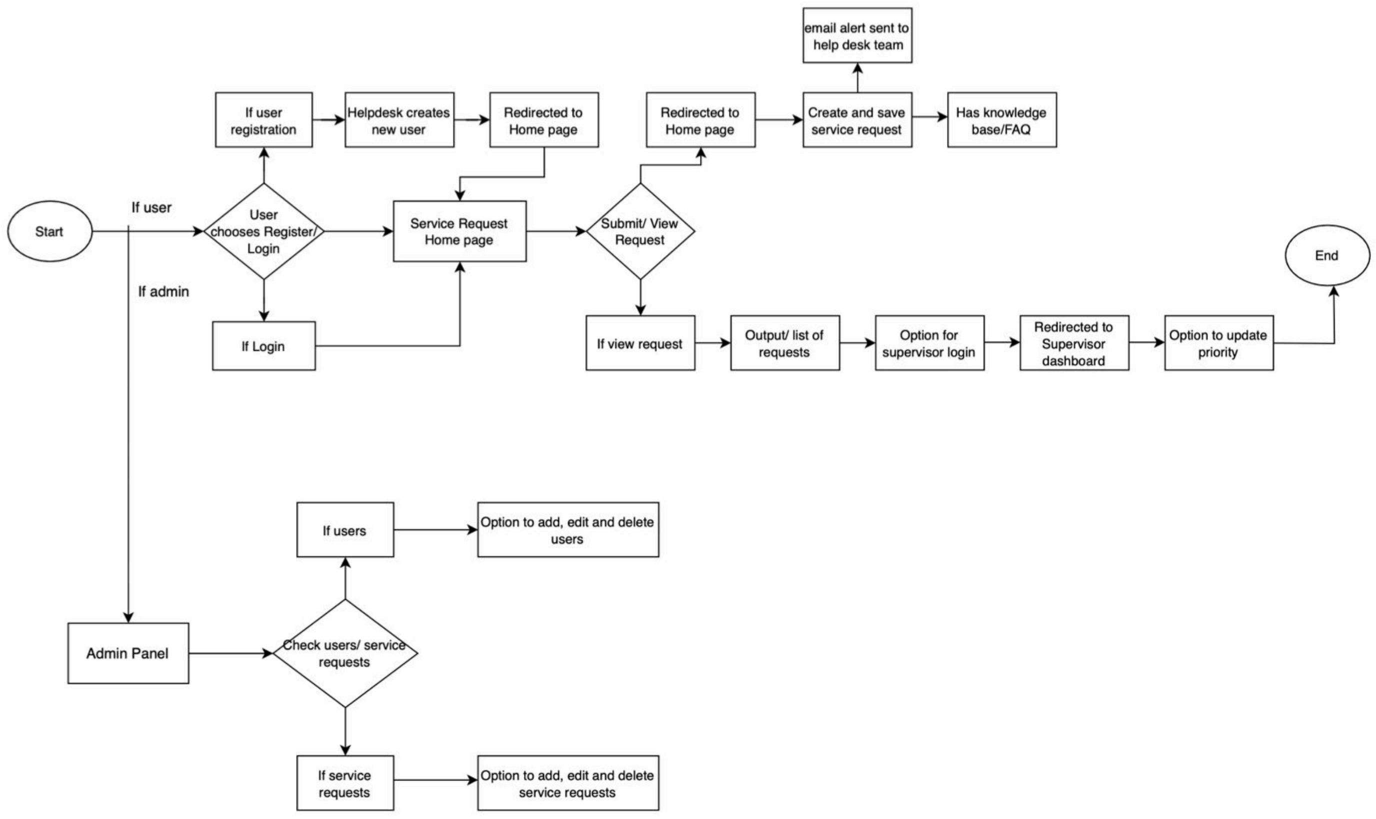
    - Admins can manage user accounts or service requests within the Admin Panel.

- **User Management:**
    - The Admin Panel provides options to add, edit, or delete users.
    - Admins can also check user service requests.
    - This facilitates the management of service request records and maintains the integrity of the helpdesk system.

UML Diagram :


User
+UserID: int
+Name: string
+Email: string
+Password: string
+Role: string
+RegisterUser(Name, Email, Role) : : string
+Login(Email, Password) : : string
1
*
ServiceRequest
+RequestID: int
+Title: string
+Description: string
+Status: string
+SubmittedBy: int
+AssignedTo: int
+SubmitRequest(UserID, Title, Description) : : string
+UpdateRequestStatus(RequestID, NewStatus, Comment, UserID) : : string
1
*
1
1
Comment
+CommentID: int
+RequestID: int
+CommentText: string
+CreatedBy: int
+CreatedAt: DateTime
+AddComment(RequestID, CommentText, UserID)
SupervisorApproval
+ApprovalID: int
+RequestID: int
+ApprovedBy: int
+ApprovalStatus: string
+ApprovalComment: string
+ApprovalDate: DateTime
+ApproveRejectRequest(RequestID, ManagerID, ApprovalStatus, ApprovalComment) : : string

Flowchart :


Start
If user
If admin
User chooses Register/ Login
If user registration
Helpdesk creates new user
Redirected to Home page
If Login
Service Request Home page
Submit/ View Request
Redirected to Home page
Create and save service request
email alert sent to help desk team
Has knowledge base/FAQ
If view request
Output/ list of requests
Option for supervisor login
Redirected to Supervisor dashboard
Option to update priority
End
Admin Panel
Check users/ service requests
If users
Option to add, edit and delete users
If service requests
Option to add, edit and delete service requests

Requirements

Functional Requirements :

- **Advanced Search and Filtering:** In service request management, the ability to filter tickets based on different criteria (severity, status, date range) is included for better manageability.
- **User Role Management:** Allow administrators to define and manage different user roles and permissions within the system for granular access control.
- **SLA Management:** Implement functionality to define, manage, and track Service Level Agreements (SLAs) for different service requests.

Non-functional Requirements :

- **Scalability:** Ensure that the system can scale horizontally to accommodate increasing service requests without manual intervention.
- Accessibility: The application should meet accessibility standards (such as WCAG) to ensure it is usable by people with disabilities.
- **Disaster Recovery:** Plan for disaster recovery to ensure system availability and data integrity in case of an outage or other catastrophic events.
- **Internationalization:** Prepare the system for multiple languages and regions, considering different time zones and locales.

Resource Allocation

- **Team Members:** Tasks were distributed based on expertise, with developers focusing on their areas of strength, such as backend, frontend, security testing, or database management.

- **Tools and Technologies:**
    - Frontend Technologies: The user interface is developed using HTML, CSS, and JavaScript. The modular file structure ensures each page and functionality is separate, facilitating more manageable maintenance and updates. This setup allows for

    dynamic content rendering and interaction with the backend via API calls.
    - Backend Technologies: The backend is powered by Node.js. It manages authentication, business logic, API endpoints, and communication with the database. The system's structure includes an entry point (index.js), environment variables (*.env), and dependency management (package.json, package-lock.json).
    - Database Technologies: The system uses Google Cloud SQL (MySQL) for database services, ensuring data persistence, integrity, query processing, and secure access. The database is structured around tables corresponding to classes such as Users and ServiceRequests.

Testing

Testing Strategy:

- **Unit Testing:**
    - Objective: To independently test the smallest units of code (functions, methods, classes) for expected behavior.
    - Tools: Utilize frameworks like Mocha and Chai for Node.js backend and Jest or Jasmine for frontend JavaScript testing.
- Integration Testing:
    - Objective: To verify the integration points between different application layers, such as the interaction between the client, server, and database.
    - Tools: Employ Postman for API integration testing and Cypress or Selenium for end-to-end workflow validations.
- User Acceptance Testing (UAT):
    - Objective: To confirm that the system fulfills the intended user needs and business goals before release.
    - Process: Engage a representative end-user group to perform tasks that reflect real-world usage, document feedback, and iterate as necessary.

Test Cases and Results:

- **User Login:**
    - Scenario: Conduct tests using a variety of valid and invalid credential combinations.
    - Outcome: Expectation of passing all test cases with correct outcomes, such as access granted for valid credentials and access denied for invalid ones.
- **Service Request Submission:**
    - Scenario: Ensure requests can be created successfully, handle failure scenarios due to incomplete fields, and test file attachment functionality.
    - Outcome: A bug was discovered in file attachment handling, which was addressed and resolved.

Debugging:

- Debugging Tools and Techniques:
    - IDE Debuggers: Utilize Visual Studio Code's powerful debugging capabilities, which support frontend JavaScript and Node.js debugging.
    - Browser DevTools: Employ browser-based tools like Chrome DevTools or Firefox Developer Tools for frontend debugging, which provide features for inspecting elements, profiling page loads, and monitoring network requests.
    - Backend Debugging: Use Node.js built-in debugging utilities and tools like node-inspector to examine backend code.
    - Logging and Monitoring: Integrate a structured logging framework such as Winston or Bunyan in Node.js, which offers various log verbosity levels and can be configured for file or console outputs. This is crucial for tracing the flow of operations and diagnosing issues in real time.
- Continuous Integration (CI) Pipeline:
    - Objective: To integrate automated testing within the CI pipeline using platforms like Jenkins or GitHub Actions.
    - Process: Set up automated tests for every code commit or pull request. Failures in this pipeline should trigger alerts and halt deployments until resolved.

Challenges and Solutions:

- **Adoption of New Technologies:**
    - Issue: The transition to new technologies can be challenging for the development team, requiring new skills and an understanding of cloud-based operations.
    - Implemented Solution: Provide comprehensive training and resources for team members.

- **Data Migration:**
    - Issue: Migrating existing data from local databases to Google Cloud SQL might present risks and technical challenges.
    - Implemented Solution: Develop a robust data migration plan with a migration test plan and rollback strategies to minimize data integrity issues.

- **Integration with Existing Systems:**
    - Issue: Ensuring the new web-based application integrates seamlessly with the users' existing systems and workflows.
    - Implemented Solution: Conduct a thorough analysis of existing systems in the development process to ensure compatibility and smooth integration.

- **Security and Compliance:**
    - Issue: Shifting to a cloud-based solution introduces new security challenges and compliance requirements.
    - Implemented Solution: Implement best-practice security measures such as rest and transit encryption, regular security tests, and compliance checks against industry standards.

- **Monitoring and Feedback:**
    - Issue: Continuously improving the application based on user feedback and system performance.
    - Implemented Solution: Integrate user feedback mechanisms directly into the application and set up monitoring tools to track usage patterns, system performance, and potential issues.