University Helpdesk CRM with Agentforce Documentation

This document provides a comprehensive overview of the implementation phases for the University Helpdesk CRM project, built on the Salesforce Developer platform.

Phase 1: Problem Understanding & Industry Analysis

- **Problem Statement**: Universities struggle with manual, delayed responses to student queries, leading to inefficient staff time and a lack of centralized issue tracking.
- **Goal**: To implement a Salesforce-native CRM that:
  - Provides 24/7 AI-powered FAQ responses.
  - Automatically creates and routes cases.
  - Enables efficient case resolution via the Service Console and Knowledge.
  - Collects student feedback after case closure.
- **Proposed Process**: A student interacts with an Experience Cloud chatbot. If unresolved, a Case is automatically created and routed to the correct department's queue.

Phase 2: Org Setup & Configuration

- **Company Profile**: Configured the org with the name "University Helpdesk CRM" and set the time zone to India Standard Time (GMT+05:30).
- **Users, Roles & Profiles**: A role hierarchy was created, including roles for "University Admin," "Helpdesk Agent," and department-specific staff. Custom profiles were created to define access permissions.
- **Queues & Case Assignment Rules**: Queues were established for IT, Finance, Exam, and Hostel departments. A Case Assignment Rule was configured to automatically route cases to the correct queue based on a Case Category picklist field.

Phase 3: Data Modeling & Relationships

- **Standard Objects**: The project utilizes standard Salesforce objects, including **Contact** (for student information), **Case** (for support tickets), and **Knowledge** (for FAQs).
- **Custom Objects**: Custom objects were created to manage specific data, including Department__c and Feedback__c.
- **Custom Fields**: Key custom fields were created to support the business process, such as a Student_ID__c field on the Contact object and a Category__c picklist on the Case object.

Phase 4: Process Automation (Admin)

- **Flows**: A Record-Triggered Flow was built to automatically create a Case when a student submits a query from the public website. This flow also calls an Apex class to enrich the

case with student details.
- **Case Assignment Rules**: Automated routing of cases to the correct department queue based on the Category__c field was implemented and configured.
- **Validation Rules**: A validation rule was created to ensure that the Student_ID__c field is filled out for new cases, enforcing data quality.

Phase 5: Apex Programming (Developer)

- **Apex Class**: The StudentInfoService Apex class was created to perform an HTTP callout to a mock REST API, demonstrating external data retrieval and JSON deserialization.
- **Apex Trigger**: An Apex Trigger was implemented on the Case object to publish a Platform Event when a case's status changes to "Escalated," showcasing event-driven architecture.
- **Test Classes**: A comprehensive test class (StudentInfoServiceTest) and a mock callout class (MockStudentInfoGenerator) were created to achieve 100% code coverage.

Phase 6: User Interface Development

- **Lightning App Builder**: The Case Record Page was customized to provide a unified view for agents, displaying the related student's information, department details, and a quick view of suggested Knowledge Articles.
- **Custom Tabs**: Custom tabs were created for easy navigation to the Feedback__c and Department__c objects.

Phase 7: Integration & External Access

- **Mock REST Integration**: A mock API was used to simulate a real-world integration, demonstrating how Salesforce can connect to external systems to fetch data.
- **Named Credentials**: A Named Credential was set up to securely manage the mock API's endpoint URL without hardcoding.
- **Platform Events**: A custom Platform Event (Case_Escalation_Event__e) was created to showcase asynchronous communication between Salesforce processes.

Phase 8: Data Management & Deployment

- **Data Import**: The Data Import Wizard was used to load sample student records from a CSV file. The process for linking cases to students using Data Loader was also prepared.
- **Data Quality**: A Duplicate Rule was configured on the Contact object to prevent the creation of duplicate student records.
- **Deployment**: The deployment strategy, using both point-and-click (Change Sets) and command-line (SFDX) methods, was defined to manage the migration of project metadata.

Phase 9: Reporting, Dashboards & Security Review

- **Reports**: Reports were created to track key metrics, such as "Open Cases by Department" and "Average Case Resolution Time," providing valuable insights into helpdesk performance.
- **Dashboards**: A dashboard was designed to provide a high-level summary for university admins, including charts on case volume and student satisfaction.
- **Security**: Field-level security was applied to sensitive fields to protect student data.

Phase 10: Final Presentation & Demo

- **Live Demo Flow**: An end-to-end demo flow was prepared to showcase the entire process, from a student submitting a query on the public website to an agent resolving the automatically created case in the Service Console.
- **Project Showcase**: The completed project and documentation are prepared for a professional presentation to showcase the implemented solution.