D424 – Software Capstone

Task 2

Capstone Proposal Project Name: Internal CRM Software (iCRM)

Student Name: Redacted

Table of Contents

Business Problem

The Customer

Chirp, Inc. (CI) is an early-stage telecommunications company headquartered in San Francisco, CA. CI specializes in providing affordable, high-speed internet and mobile services to underserved urban and rural communities. The company is focused on scalability, customer satisfaction, and service reliability as it aims to compete with larger telecom providers. Their mission is to provide fast, reliable, and affordable telecommunications services while maintaining outstanding customer satisfaction through efficient service management.

The size of the company is currently 50 employees and growing. The company projects to scale to 200 employees within 3 years. The company currently has approximately 5000 subscribers, with an anticipated increase to 50,000 in five years.

The company includes various teams including customer service, technical support, customer success managers, and operations teams. Amongst these teams include customer support agents, who directly work with individual customer issues and resolve service tickets, customer success managers (CSMs) who ensure customer satisfaction, and operations analysts, who manage the infrastructure, databases, and internal systems of the company.

Currently, Chirp's customer support team relies on spreadsheets and manual logging, leading to lost or misplaced cases, slow response times, no visible case history for repeating customers, and difficulties prioritizing cases.

Chirp's short-term goals include implementing an internal CRM for customer service cases to streamline their case resolution process. A key metric they are tracking is case resolution time, which they aim to reduce by 40% to improve customer satisfaction. They want to establish a good software and hardware infrastructure to support the company's growth as they look to scale nationwide and compete against the powerhouses of the Telecommunications domain.

Chirp's long-term goals include expanding the company's presence throughout the United States as well as nationwide. They wish to compete with the likes of other popular telecom companies like Verizon, Sprint, etc. They want to be a more sustainable and socially responsible telecommunications leader by providing low-cost/free connectivity programs for students and non-profits. They want to stay ahead of the competition by incorporating next-generation tooling and technology.

Business Case

iCRM will help Chirp by standardizing case management, improving customer satisfaction, cutting down case response times, and providing a simple yet scalable solution that will help Chirp grow while expanding its client base.

Chirp is currently struggling to scale because there is no centralized case management system in place. As customers call in and open new service cases, each customer service agent has their own personalized way of logging the case, and data is saved on each agent's personal machine. Agents rely on spreadsheets and manual logging to track incoming and outgoing cases, leading to lost or misplaced cases, slow response times, no visible case history for repeating customers, and difficulties prioritizing cases.

As it is currently, Chirp lacks organization, accountability, and scalability. As the business grows and customers open more service cases, the process of each member tracking their own individual cases can become overwhelming. Cases can go untouched for long periods of time if there is no system in place to hold employees accountable for resolving cases. Customer details can be scattered across emails, spreadsheets, and other note-taking mediums. Employees waste time asking for updates on internal cases or searching for past interactions.

iCRM will solve these challenges by providing a centralized case management system that will address these organizational issues. iCRM will make it easier to manage customer interactions, resulting in improved company branding and increasing customer satisfaction or CSAT (Customer Satisfaction Score). iCRM will provide a single, multi-purpose, comprehensive platform that helps employees track, search, and manage cases, interactions, priorities, and customers.

By incorporating iCRM, Chirp will greatly improve case resolution times by streamlining workflows, enhance customer satisfaction by providing structured follow-ups, and increase operational efficiency by centralizing all case data.

Fulfillment

iCRM aims to be an all-in-one software solution for meeting the internal needs of Chirp, including the customer service team and all other internal teams. The software will be accessible to all Chirp employees through a web browser. Employees will be authenticated and authorized to adhere to security principles. Users will have access to a modern, sleek, and intuitive user interface, making the process of tracking cases quick and painless.

iCRM will feature a tabular dashboard that display relevant business information, like Cases, Contacts, Businesses, etc. Users will be able to export this information into CSV format as a report to share at their discretion. Users can also create, update, and maintain various entities like Cases and Contacts to store essential business data. Users will be able to sort and filter cases by any metric or field that they choose; an example is if a user wants to work on their highest priority cases first, they will sort by descending priority.

Users will also be able to collaborate on cases through case comments, with each case containing a feed for that case. This will increase inter-team communication, which is a good sign for any well-functioning business.

SDLC Methodology

Because we're developing an MVP-level product to start, we're focusing on fast delivery, and Agile Scrum is the project methodology that would work best for developing iCRM. Agile Scrum is an iterative approach to developing software projects that focuses on continuous releases and weaving feedback from customers into each iteration. The Agile Scrum methodology is ideal for this project because it emphasizes adaptability, rapid iteration, and continuous feedback. Since the CRM is being built as an MVP within a limited timeframe, Scrum allows the development team to focus on delivering high-priority features first while remaining flexible to changes based on real-time feedback from Chirp's internal users. The iterative nature of Scrum ensures that functionality is tested and refined in small

increments, reducing the risk of large-scale rework. Additionally, frequent check-ins and reviews promote collaboration between developers and stakeholders, ensuring that the final product aligns closely with the company's needs. By breaking down development into short, manageable sprints, Agile Scrum facilitates efficient progress tracking. It allows for quick adjustments if new requirements emerge, ultimately leading to a more polished and user-centric CRM.

There are multiple development phases in this agile scrum process. Phase 1 is product backlog creation, in which the team will identify the core functionalities of the CRM, examples being case management and search capabilities. In this phase, the team will gather requirements from stakeholders to define the most critical features for the MVP.

Phase 2 is Sprint Planning, a phase in which the team breaks these identified features into chunks called "User Stories". These sprint plans help set the team up for success throughout the sprint by defining acceptance criteria for each feature, ensuring clarity on when the sprint or task is considered "done."

Phase 3 is Sprint Execution, a phase in which the development process begins. In this phase, the team works in short bursts of 1-2 weeks to implement a specific module or ticket. In this phase, a daily stand-up meeting occurs that gives the team members opportunities to review progress, identify blockers, and adjust priorities if needed. Developers update their team here and discuss any challenges requiring teammate help.

Phase 4 is Sprint Review & Retrospective, a phase in which the team showcases the completed work to stakeholders for feedback at the end of the sprint. In this phase, feedback is received, and necessary adjustments are noted to the team. The team will evaluate what went well and what could be done better. Technical challenges will be discussed in this phase, as well as backlog adjustments.

After this phase, the entire pattern repeats itself, which is called Iteration and Continuous Improvement. The team continues to adjust the backlog, plan sprints, execute those sprints, and sprint review to improve and tailor the software toward customer needs.

Deliverables

1) Product Backlog Document:
    a. A document describing the list of features, user stories, and future enhancements needed for the CRM.
    b. Example: A Trello board containing user stories like, "As a customer support agent, I want to search for cases by customer name so I can quickly locate relevant issues."
2) User Stories & Acceptance Criteria:
    a. User Stories: Clear, structured descriptions of CRM functionalities with expected outcomes.
    b. Acceptance Criteria: The set of conditions or outcomes that a story must meet to be considered complete.
    c. Example: "The case owner can be changed via a dropdown in the user interface."
3) Wireframes & Design Mockups
    a. These are visual representations of the CRM's interface. They are low-fidelity mockups

4) High Fidelity Prototype
    a. Once the wireframe is polished and passed through several rounds of approval and iteration, a higher fidelity prototype is created that developers can reference while creating the application's front end. This prototype is usually created in Figma or Canva and contains front-end user flows from screen to screen.
5) Functional MVP
    a. This is a working prototype of the CRM with implemented core functionalities like user authentication, case management, and search. This prototype will be deployed and has a functional codebase.
    b. Example: A deployed and functional web application with a login system, a CRM dashboard displaying open cases, and a working case creation and assignment system.
6) Source Code
    a. A GitHub repository with version-controlled code engineered with scalability and maintainability in mind.
    b. Example: A GitHub repository with organized folders: "/frontend", "/backend", "/database"
7) Test Suite
    a. Documentation surrounding test cases, as well as a suite of tests that ensure the app is always working as intended.
    b. Example: Jest test cases for React & Node.js and a README.md file containing test strategies, plans, pass/fail criteria, and expected results.
8) Deployment & Release Notes
    a. Documentation detailing how to deploy the CRM, including dependencies, configurations, and version changes.
    b. Example: README.md with setup instructions, and release notes outlining new features, bugfixes, and known issues

Deployment Plan and Outcomes

Chirp's Internal CRM (iCRM) is a customer case management system designed for internal use by Chirp's customer service representatives. This web application allows employees to log, track, update, and resolve customer inquiries by creating, updating, and logging cases alongside customers and their contact information. The system results in faster case turnaround times and increased customer satisfaction via case searching and streamlined employee workflows.

The following is a deployment plan meant as a guide for employees and stakeholders to deploy iCRM to a production environment. Outcomes from this deployment include a live web application meant to support Chirp's customer service team to increase case turnaround time and help Chirp adhere to client Service-Level Agreements. Technical documentation will also be produced as a deliverable, examples being system architecture guides, API guides, UML diagrams, and database connectivity guides. User training manuals will also be produced to introduce new customer service agents to iCRM. Video guides/screen recordings will be hosted on Chirp's training materials SharePoint to allow for

asynchronous training. Developers will host live sessions to initially train team members on how to use iCRM as well.

The development plan will be as follows:

Pre-deployment:

Before launching the CRM, IT administrators and developers must ensure that the proper cloud infrastructure is in place to support the deployment. A production environment must be created on Railway. This environment will contain the code for a NestJS backend, and a singular PostgreSQL database server. The frontend code will be hosted on Vercel, so infrastructure must also be provisioned on this cloud platform. Developers must work alongside the QA team to run any functional, integration, and unit tests on the testing branch to verify that the software works as intended.

Deployment:

Developers will deploy the backend services (backend and database) using cloud provider Railway. They will also configure the PostgreSQL database and set production database credentials, API keys, and other authentication secrets. Once complete, Database Administrators will verify that there are no security breaches and will verify that sensitive information is being handled correctly.

Once the backend is set up, developers will host the front end of iCRM on Vercel. Connectivity between the application's front and back end will be automatically tested during deployment via a separate Continuous Integration & Continuous Delivery pipeline that QA engineers will have already set up. IT Admins will then ensure that HTTPS connectivity and all provisioned cloud infrastructure meet security needs and that all web traffic is secured throughout the app.

After connectivity is secured, test data will be seeded into the application, and the final end-to-end tests will be conducted. These tests will both be done automatically via an end-to-end testing suite like Playwright and manually via the QA team. Any bugs will be reported and sent to the development team and will be logged and added to the developer backlog, ready to be fixed by the next production cycle.

Post-Deployment

After the software is deployed, customer service agents and any users of the app will have access to documentation hosted on Chirp's internal knowledge base including user guides, FAQs, and troubleshooting tips. Users will be able to generate IT Support tickets if they need assistance using the software or if they want to put in feature requests for future iterations of the software.

Project Timeline

Phase	Milestone/Task	Resources	Dependencies	Deliverable	Description	Dates w/ Duration
Planning	Define project scope & purpose	Project Lead	Business case & problem definition	Business requirements document (BRD), the purpose of the	Defining software scope. Answers the question,	2/18/2025 – 2/22/2025 5 days

				app is well-defined	"What problem is this software solving".	
Requirements Analysis	Identify requirements of the final users	Project Lead, Developers, Stakeholders	Business requirement document, approved project plan	User Stories, product backlog document, Functional & Non-functional requirements, Software Requirements Specification	Gathering and validation of necessary requirements before design of software. Verify what exact requirements need to be fulfilled by the software	2/23/2025 – 2/28/2025 6 days
Design	Create design documents	UI/UX Team, Developers	Stakeholder requirements approval, completed Software Requirements Specification	wireframes, UI mockups, high-fidelity prototypes, entity-relationship diagrams, software design document	Define software structure, UI, entity relationships to provide structure and create a roadmap for developers	3/1/2025 - 3/7/2025 7 days
Development	Conduct development sprint cycles to finish functional MVP	Developers, Development Team Lead	A completed Software Design Document, alongside UI wireframes, prototypes, entity-relationship diagram	MVP prototype of the software application	Implement core functionalities of the app according to the software design document. Conduct code reviews.	3/8/2025 – 4/5/2025 4 weeks
Testing	Create & execute tests	QA Team, Developers	A completed development cycle/MVP software	A completed test suite containing unit tests, integration tests, e2e tests, and documentation	Conduct various types of tests to examine software for bugs/glitches to ensure software operation	4/6/2025 - 4/13/2025 8 days

				surrounding these tests		
Deployment	Deploy software & document the process	Developers, Development Team Lead	An MVP with completed minimum functionality that fulfills shareholder purpose	A production-level deployed application ready for end-users to use. Deployment Plan document.	Software product rollout for end-users. Users can provide feedback on the software and learn how to pilot the software using documentation created in this process	4/14/2025 - 4/15/2025 2 days
Maintenance	Monitor system performance, plan for future improvements	Developers, QA, Project Lead	Successful production deployment, comprehensive technical documentation	Release notes, updated documentation, system updates	Ensure software longevity by monitoring and adjusting to customer expectations. Refine and adapt to changing stakeholder needs.	4/16/2025 – Ongoing Ongoing

Environments and Costs

Programming Environment

Hardware Requirements

- Developer Workstations: Each team member will require a computer with adequate processing power, memory, and storage capacity to handle development tasks efficiently. A reliable internet connection is necessary for accessing cloud services, version control, and remote databases.

Software Requirements

Backend Technologies (Node.js & NestJS): The backend will be powered by NestJS, a Node.js framework that utilizes DI (Dependency Injection) designed for scalability and maintainability. Developers should install the latest stable version of Node.js and the NestJS CLI for efficient project setup and management.

Frontend Technologies (React & Tailwind CSS): The user interface will be built with React, providing a dynamic and interactive experience. Tailwind CSS will be used for styling, allowing for rapid UI development. Developers must configure a React environment with the necessary Tailwind setup.

Frontend Build tool (Vite): Modern frontend build tool that compiles, bundles, and transpiles JavaScript/TypeScript code.

Database (PostgreSQL): PostgreSQL will serve as the primary database for storing customer data and case management information. A database instance should be set up both locally for development and on a cloud service for production.

Code Editor & Development Tools: Visual Studio Code (VS Code) is recommended for writing and managing code. Extensions such as ESLint, Prettier, and Tailwind IntelliSense should be installed to enhance code quality and development efficiency.

Version Control & Collaboration: Git and GitHub will be used for source control and team collaboration. Team members should follow a structured branching strategy and submit changes via pull requests.

Containerization & Deployment: Docker may be utilized for creating a consistent development environment. The application will be deployed on a cloud provider such as AWS or Railway.

Testing & Debugging Tools: Jest will be used for unit and integration testing. NestJS includes built-in testing integration with Jest. Postman will facilitate API testing. React Devtools will help in debugging react components alongside state management.

Browser: An updated, modern browser like Chrome or Firefox will be needed for testing and development.

Environment Costs

Initial Costs:

- Development Machines (Work laptops) - $1,500 per developer. Ensures that developers have powerful hardware capable of running development environments, databases, and testing frameworks efficiently.

Recurring Costs:

- Domain registration through SquareSpace is $12/year. Vercel provides a free, uncustomizable domain, but we choose to pay for the professional aspect of having a custom domain.
- Railway Cloud NestJS/Database Backend Environment: The Railway “Pro” Plan costs $240/year per developer on the project. This includes 32GB RAM / 32 vCPU per service. The estimated cost with 5 developers on the project per year comes out to $1,200 not including excess usage fees. The backend API and PostgreSQL databases can run in this environment.
- Vercel Cloud Frontend Environment: The Vercel “Pro” Plan costs $240/year per developer on the project. It comes with increased usage limits compared to the “Hobby” plan as well. The estimated cost with 5 developers on the project per year comes out to $1,200, not including excess usage fees. The React frontend will be hosted on this platform.

- Project Management Tools: Jira Premium costs around $100/user per year. This subscription is needed for task tracking, collaboration, and communication between developers. Total cost is $500/year for a development team of 5.

Human Resource Requirements

To complete the Chirp Internal CRM MVP, a team of five developers will be working 40 hours each at an assumed hourly rate of $60/hour.

1. Calculating Labor Cost per Developer

Labor cost per person = Hourly rate × Total hours worked

Labor cost per person = $60 × 40

Labor cost per person = $2,400

2. Calculating Total Labor Cost

Total labor cost = Labor cost per person × Number of developers

Total labor cost = $2,400 × 5

Total labor cost = $12,000

Thus, the total labor cost for the Chirp Internal CRM MVP development is $12,000, assuming each developer contributes 40 hours to the project.

Validation and Verification

A structured testing approach will be implemented to ensure the Chirp Internal CRM MVP is reliable, functional, and user-friendly. This process will be cyclical, involving continuous iteration between development, testing, and bug fixing.

Unit Testing: To ensure individual components and functions work correctly, developers write unit tests using Jest (for backend) and React Testing Library (for frontend). These tests cover API endpoints, database queries, and UI interactions. Developers can either manually test before merging code into the main branch, or the QA team can develop a CI/CD pipeline to run a test suite before every attempted merge automatically. Code must pass all unit tests before proceeding. This process prevents small bugs from compounding into larger issues later in development.

Integration & End-to-end Testing: Playwright will be used for automated integration and end-to-end (e2e) testing after merging to the staging branch. These tests ensure proper communication between the frontend, backend, and database and ensure that different parts of the system work seamlessly

before moving to production. Automated simulated user interactions via Playwright/Selenium will verify that data flows correctly across the system during common user flows. This process validates that multiple modules or components function correctly together and that the website functions similarly to how users would expect. Both developers and QA engineers will conduct this process of testing.

User Acceptance Testing (UAT): Once the code passes necessary integration & e2e tests, the next step is to conduct UAT to ensure the system meets business needs and user expectations. These tests are conducted by business analysts & selected users (Client Representatives). In this stage, predefined test scenarios are shared with users to validate key features, such as case creation, filtering, sorting, and report generation. Selected users will provide feedback on the application's usability, functionality, and performance. All necessary changes are to be implemented before deployment. This stage of testing guarantees that the software is usable and meets client expectations.

Bug Tracking & Resolution Process:

When bugs are identified in the system, the issues are logged in Jira and added to the project backlog for developers to tackle. Each bug is assigned a priority (Critical, High, Medium, Low). The project manager or team lead assigns the bug to a developer based on developer expertise. The developer fixes the issue in a feature branch, and their code undergoes peer review before merging. Meanwhile, the QA team will then create new unit, integration, and e2e tests for the new code to pass. The bugfix will then run in the CI/CD pipeline, running the test suites listed above to test if the issue has been resolved. Once approved, the fix is merged into the main branch and deployed to production.

Testing is considered complete when all test suites pass successfully and Chirp confirms that the system meets their business needs. This approach ensures Chirp’s Internal CRM is stable, secure, and functional with each successful deployment.