

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

TEXAS A&M UNIVERSITY, COLLEGE STATION



CSCE 606: SOFTWARE ENGINEERING

CRM SERVICE FINAL REPORT

Team Members and Roles

Jahn timer Anugu
Anu Deepika Devara
Sahithi Ravipati
Varshani Reddy Patlolla

Scrum Master & Project Manager (On rotation)

Iteration0: Sahithi, Varshani
Iteration1: Anu Deepika, Jahn timer
Iteration2: Varshani, Sahithi
Iteration3: Jahn timer, Anu Deepika
Iteration4: Sahithi, Varshani
Iteration5: Anu Deepika, Jahn timer

Summary:

As we all know Customer satisfaction is the goal of every organization. Earlier everything used to be done manually using excel sheets or papers, but with increasing data, it would become a tedious task. Then there was a requirement to store and analyze the data in an effective manner to improve customer satisfaction. It is when CRMs were created. A CRM Service allows us to track customer activities, manage emails, social media, and marketing, analyse obtained customer data, and much more.

As a part of this project, we have implemented an initial level of CRM Service, which connects two applications CAST.NXT and EVENT.NXT of the FASHION.NXT industry. We fetch customer-related information from these applications and allow a CRM user to access this data, perform some analytics on how frequently their pages are being visited and send communication emails to the user. We have built this application using Ruby on Rails, JavaScript, HTML and CSS. Integrated with PostgreSQL database in the development environment and SQLite in the production environment. We have used GitHub for storing our code files and tracking the implementations. Our application is deployed on Heroku. The story points of the project are monitored using the Pivotal tracker. The stakeholder of this project is Tito Chowdhury. He hasn't specified any strong user stories at the beginning of this project, but in the later stages, he gave some CRM-specific requirements, which we have implemented (fetching data from CAST.NXT, restricting access to CAST.NXT users using our CRM service), although we could not integrate with EVENT.NXT team, as they didn't have a sorted database structure for our collaboration, and we have listed it in future scope.

User Stories:

Initial Project Setup:

- Rails application setup
- Creation of rails DB and schema
- Heroku Setup and deployment
- Connection to a database.

Basic application functionalities:

- Login
- SignUp
- LogOut
- Creating an Admin Dashboard.

Implemented CRM core Features and previous features enhancements:

- Analytics Dashboard
- CSV Uploader
- Started implementing email feature

Login Page enhancements
Other UI page enhancements

CRM Features enhancements:

Analytics Dashboard enhancement
Data Filter feature in User Management page
Sending emails to a single user with fixed email content
Reports Downloader

Integrating with CAST.NXT applications and a few other enhancements to the application:

Secured our application internal pages(Can view only if logged-in)
Error messages on UI
API for fetching CAST.NXT users
API for making users active/inactive.
Toggle button in CRM User Management for activating/inactivating users
Email notifications for multiple users with custom email content.

Iterations:

Iteration 0:

- Organized the team and developed a meeting schedule
- Connected with the client to understand the project requirements and goals.
- Listed down a few user stories and submitted a design document
- Acquainted ourselves with the technologies being used.

Iteration 1:

- In this iteration, we worked on understanding how to build our CRM Service as it is a green field application.
- By using some online resources, we figured out our approach and started building the application.
- We have completed a basic ruby on rails application setup.
- Connected our application with a database using a random schema.
- Established the Heroku setup and deployed the application to Heroku.

Iteration 2:

- As a part of this iteration, we have implemented the standard application features like Login, Register and Logout options.
- Modified Database Schema for supporting the above login and register functionalities.
- Created a basic Admin Dashboard page which contains links to all other CRM Service Pages.

Iteration 3:

- In this iteration, we have started implementing the core CRM features below.
- Analytics Dashboard to view which pages have been visited frequently.
- Sending email communication to CAST.NXT and EVENT.NXT customers.
- A feature to upload user's data in CSV format.
- Enhanced the Login page(Added 3D Animation background, Separated Login, Registration features) and other UI pages.

Iteration 4:

- For this iteration, we have enhanced the CRM core functionalities implemented in previous iterations.
- We have added a data filter option for all the fields displayed on the user management page.
- We have implemented the email communication feature halfway, like sending emails to a single user with fixed content.
- Enhanced the analytics dashboard using Ahoy.

Iteration 5:

- As a part of this iteration, we have connected with the CAST.NXT teams and EVENT.NXT teams for getting their data to our CRM Service to be displayed.
- We have implemented an API to fetch user data from their databases(Only the CAST.NXT database was ready so that we could build a pipeline, Hence we have implemented this only with CAST.NXT)
- We have also implemented an important feature to restrict customer access to the CAST.NXT application from our CRM Service.
- Added a security feature where people not logged in to the CRM application cannot view any internal pages.
- Added Error messages on the UI whenever an unauthorized action is performed in our application.

Customer Interaction:

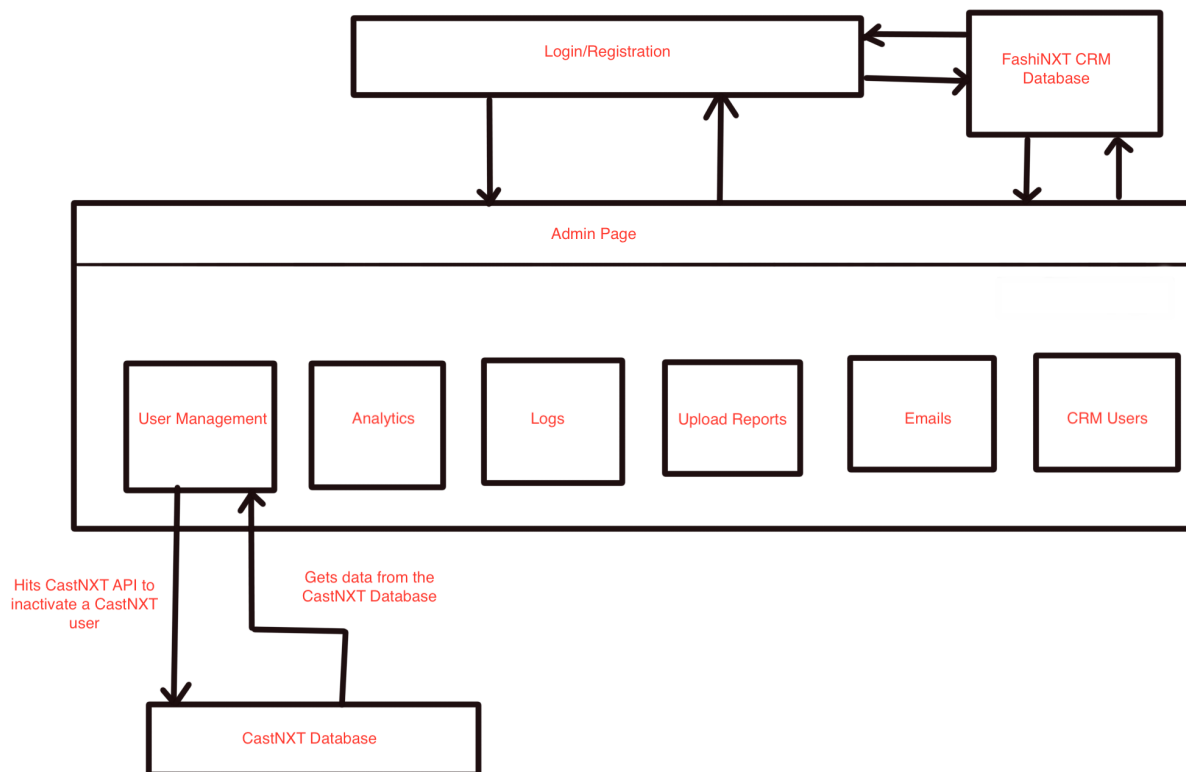
We had a zoom meeting with our client Tito Chowdhury on the below days for every Iteration.

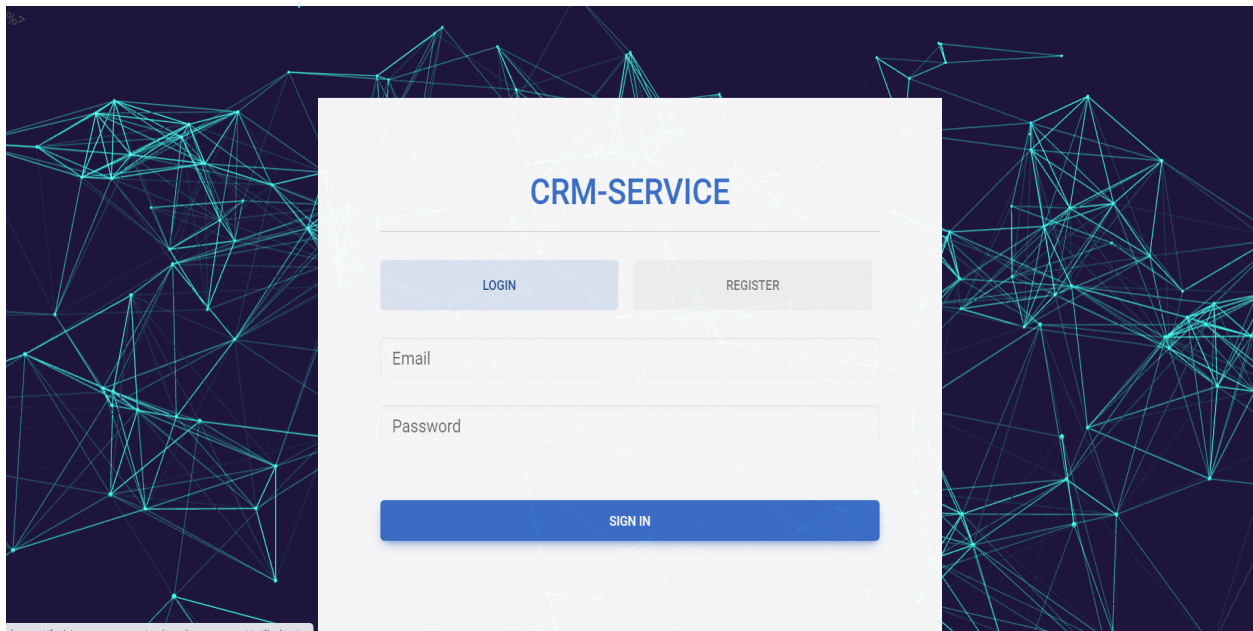
- Customer meeting 1: Connected with our client on 21st September before Iteration 0 to understand the project requirements and goals.
- Customer meeting 2: Connected with our customer on 28th September before Iteration 1 to clarify user stories that need to be implemented to build CRM Service.
- Customer meeting 3: Connected with the customer on 21st October before Iteration 2, in which we discussed our progress, demonstrated the changes (login, register, logout and admin dashboard), took some feedback, and discussed the future stories.
- Customer meeting 4: We connected with the client on November 3rd before Iteration 3 and demonstrated the core CRM features, which we implemented till then(Analytics

dashboard, user management filters, email communication, CSV uploader). He asked us to connect with the CAST.NXT and EVENT.NXT team to build a pipeline to fetch data from them.

- Customer meeting 5: We connected with the client on December 1st to demonstrate our features implemented after collaborating with CAST.NXT team(Marking users of CAST.NXT active/inactive using our CRM Service). We conveyed to him that EVENT.NXT wasn't storing any relevant data that we could integrate.
- Customer meeting 6: We connected with Tito Chowdhary on December 9th to show our final application after integrating with CAST.NXT team. As EVENT.NXT didn't have data he asked us if we could help them after the end of the semester.

Design:



Login Page:

The login page features a dark blue background with a glowing cyan network pattern. A white rectangular form is centered, containing the title "CRM-SERVICE" in blue. Below the title are two buttons: "LOGIN" (light blue) and "REGISTER" (light gray). Underneath are input fields for "Email" and "Password". At the bottom is a large blue "SIGN IN" button.

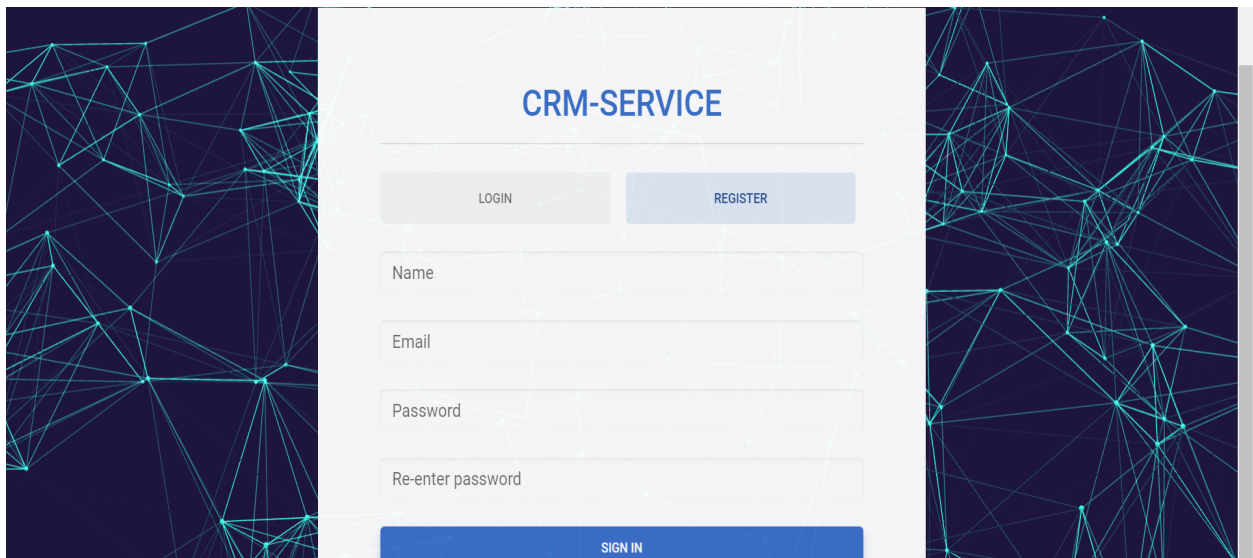
CRM-SERVICE

LOGIN REGISTER

Email

Password

SIGN IN

Registration page:

The registration page has the same dark blue background and cyan network pattern. The white form is centered with the title "CRM-SERVICE" in blue. It features two buttons: "LOGIN" (light gray) and "REGISTER" (light blue). Below are input fields for "Name", "Email", "Password", and "Re-enter password". At the bottom is a large blue "SIGN IN" button.

CRM-SERVICE

LOGIN REGISTER

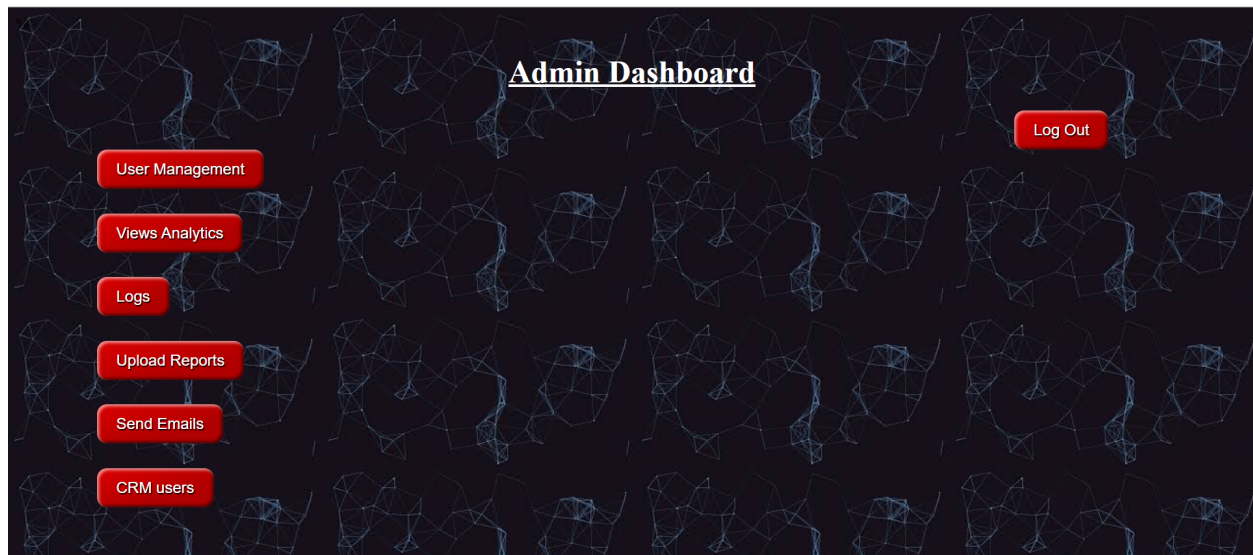
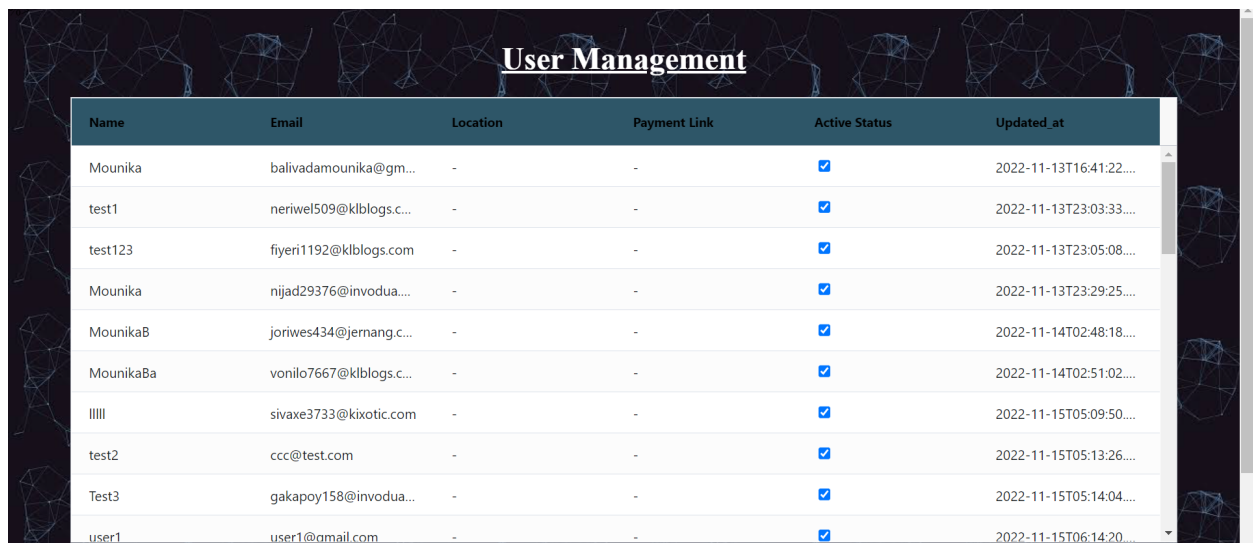
Name

Email

Password

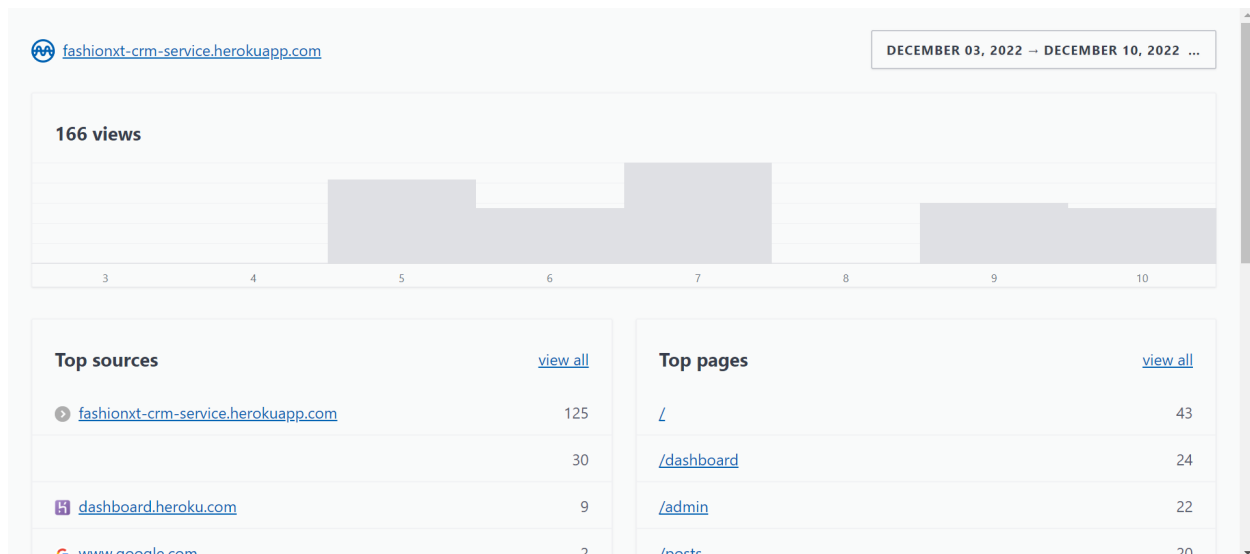
Re-enter password

SIGN IN

Admin Dashboard:User Management:

The User Management interface shows a table with user data. The title "User Management" is centered at the top in a white, underlined font. The table has six columns: Name, Email, Location, Payment Link, Active Status, and Updated_at. It contains ten rows of user data, all with an active status indicated by a blue checkmark.

Name	Email	Location	Payment Link	Active Status	Updated_at
Mounika	balivadamounika@gm...	-	-	<input checked="" type="checkbox"/>	2022-11-13T16:41:22...
test1	neriwe1509@klblogs.c...	-	-	<input checked="" type="checkbox"/>	2022-11-13T23:03:33...
test123	fiyeri1192@klblogs.com	-	-	<input checked="" type="checkbox"/>	2022-11-13T23:05:08...
Mounika	nijad29376@invodua...	-	-	<input checked="" type="checkbox"/>	2022-11-13T23:29:25...
MounikaB	joriwes434@jernang.c...	-	-	<input checked="" type="checkbox"/>	2022-11-14T02:48:18...
MounikaBa	vonilo7667@klblogs.c...	-	-	<input checked="" type="checkbox"/>	2022-11-14T02:51:02...
lllll	sivaxe3733@kixotic.com	-	-	<input checked="" type="checkbox"/>	2022-11-15T05:09:50...
test2	ccc@test.com	-	-	<input checked="" type="checkbox"/>	2022-11-15T05:13:26...
Test3	gakapoy158@invodua...	-	-	<input checked="" type="checkbox"/>	2022-11-15T05:14:04...
user1	user1@gmail.com	-	-	<input checked="" type="checkbox"/>	2022-11-15T06:14:20...

Analytics Dashboard:CSV Uploader:

Upload Reports

Name	Manager	Status	Terms			
Anu	Tito	Active	1	Show	Edit	Destroy

Import Reports

No file chosen

[New Report](#)

Send Email Communication/pdf download report:

Email Notifications						
Id	Title	Content				
1	Report_test	Contains P&L data	PDF	Show	Edit	Destroy
2	kk	kk	PDF	Show	Edit	Destroy
3	test mails	test test test	PDF	Show	Edit	Destroy
4	testing mails 2	mails	PDF	Show	Edit	Destroy
5	test mails j	testing mails by jahnavi	PDF	Show	Edit	Destroy
6	test mails 3	test	PDF	Show	Edit	Destroy

Testing:

We developed cucumber test cases for the BDD processes. And also wrote integration test cases for each controller and helper classes that we made. Rspec test cases are used for unit testing.

To run the cucumber test cases use the below command:

```
cucumber feature/<fileName.feature>
```

```
When('I go to the login page') do
  visit '/'
end

When('I fill in the following:') do |table|
  When %[I go to the sign in page]
  And %[I fill in "Email" with "#{Email}"]
  And %[I fill in "Password" with "#{Password}"]
  And %[I press "Log Me In"]
end

When('I press {string}') do |string|
  page.has_content?(text)
end
```

```
Feature: Login feature
  As an existing Cucumber user
  I want to create an account
```

```
Scenario: Login
  Given a valid user
  When I go to the login page
  And I fill in the following:
    |Email|xyz@gmail.com|
    |Password|xyz|
  And I press "Login"
  Then I should see "Admin Dashboard"
```

To run the integration tests , use the command : `bin/rails test`

Integration tests are defined in the test folder.

```

class CompaniesControllerTest < ActionController::TestCase
  setup do
    @company = companies(:one)
  end

  test "should get index" do
    get '/companies'
    assert_response :success
    assert_not_nil assigns(:companies)
  end

  test "should get new" do
    get :new
    assert_response :success
  end

  test "should create company" do
    assert_difference('Company.count') do
      post :create, params: { company: { manager: @company.manager, name: @company.name, status: @company.status, terms: @company.terms }}
    end

    assert_redirected_to company_path(assigns(:company))
  end

  test "should show company" do
    get :show, params: { id: @company }
    assert_response :success
  end

  test "should get edit" do
    get :edit, params: { id: @company }
    assert_response :success
  end
end

```

For unit test cases, we used rspec. To run them use the command : `rspec spec/models/<spec file name>.rb`
`rspec spec/models/<spec file name>.rb`

Repo Contents












For every new iteration we create a new branch from master and push all our changes to it. After completing all the changes and testing locally, the scrum master raises a PR to the master. At the end of each iteration, we merge it to master and deploy it as the “production branch”. We created a new tag for every release at the end of each iteration with the “iterationNo.” and associated it with the releases as well. We used the initial postgres db for login and for rest all data we use castNXT and eventNXT databases.

Iteration5

 5 days ago  ea719ea  zip  tar.gz

Iteration4

 17 days ago  3ea2ff8  zip  tar.gz

Default branch	
 master	Updated 29 minutes ago by sahithi-r
Your branches	
 test_cases_added_for_cucumber	Updated 18 minutes ago by anudeepikavz1
 cucumber_test_cases	Updated 5 hours ago by varshanipreddy
 itr5_testcases	Updated 5 days ago by anudeepikavzf03-tamu
 html_changes_pdf	Updated 28 days ago by anudeepikavzf03-tamu
 deployment_issues	Updated 29 days ago by anudeepikavzf03-tamu
Active branches	
 ChangeVersionPDF	Updated 5 minutes ago by sahithi-r
 test_cases_added_for_cucumber	Updated 18 minutes ago by anudeepikavz1
 cucumber_test_cases	Updated 5 hours ago by varshanipreddy
 dev	Updated 5 hours ago by j-reddy4
 mails	Updated 6 hours ago by varshanipreddy

Local Setup:

1. Clone the application through github using the url
<https://github.com/j-reddy4/FashionNXT-CRM-Service>
2. Make sure bundler is installed, else use the command:
gem install bundler
3. To install all the required gems run the command: bundle install
4. To run the db migrations : rake db:create
rake db:migrate //run this for the first time and when database model is changed
5. To run the server: rails s
6. Visit : <http://127.0.0.1:3000/>

Heroku Deployment:

We have deployed our application on Heroku

1. Deployment guide for rails application :
<https://devcenter.heroku.com/articles/getting-started-with-rails6>
2. Procfile is required for Heroku deployment. Create a Procfile with the below data in it
web: bundle exec puma -t 5:5 -p
PORT:-3000-e
{RACK_ENV:-development}
3. After pushing the changes to Github, Run db migrations:

```
heroku run rake db:migrate
```

Note: Migrations are to be run for the first time and only when there's a change in Database models

4. Make your changes live on Heroku using the command:

```
git push heroku main
```

Issues faced :

1. There may occasionally be a discrepancy between your local repository and the heroku remote repository. To resolve this, you need to either pull the heroku remote repository and push your changes OR do a git force push.
2. Sometimes, there might be deployment errors or a feature might not work in heroku but works locally. For this go through the logs using the command : `heroku logs -tail` and check the logs for any errors or missing files. Resolving these issues will resolve the problem.

Issues/ Challenges

1. The major issue was that the client was unclear of his requirements in the starting phase. This delayed our planning process, and requirements were changed frequently and no clear explanation was given. We had to redo many features because of the communication gaps.
2. The next major challenge was the integration of databases of other teams with our application. Since the team castNXT used mongodb, we initially faced the CORS policy issue.
3. We used wicked_pdf gem for the development of pdf downloader and this is a very unstable gem in the production environment. This has to be kept up to date.
4. We had to display the analytics of another team's application via our CRM application, injecting the feature to another's repository was a challenge as we had to go through their code base, learn about their processes , methods, and gems.
5. Git merge conflicts were hard to resolve when one updates the binary file in the stream and someone already deleted that file in the upstream.

Gems Used

1. Wicked_pdf : this was used for the pdf downloader. This converts the html view created by us as a pdf.
2. Rspec, rspec-rails, cucumber-rails, selenium-webdrivers,rail-controller-testing : All these gems are added for testing purposes. They are used in the TDD/BDD process
3. Ahoy_matey :This gem is used to track events and visits and the data is automatically stored to the local database. This was very helpful for the analytics part where we needed to track the users data. Although this sits only in the backend at the moment it can be used in the CastNXT and EventNXT applications and results can be published in the CRM

4. `Active_analytics`: This gem helped generate the view count for the application and the pages within it.
5. `Net-smtp`, `net-imap`, `net-pop`: These are used for our email services to provide functionality to send mails via SMTP calls. At present these are not used as we are sending mails over the server.

Software Development Process:

We have followed the BDD methodology. Following BDD has helped speed up the development process. With BDD, as its name suggests, the focus is on behavior rather than implementation itself. BDD enhances the quality of the code, thus reducing the maintenance cost and minimizing project risk.

We also followed agile methodology, where we held weekly scrum meetings to track our progress. A sprint planning meeting was held biweekly to discuss the jira tickets we will be working on. The developer writes the unit test cases and an integration test is conducted at the end of feature development. We would create tickets in the current iteration and if any new ticket is added, we would push it to “backlog” and pick it up in the next iteration based on priority. We used a pivotal tracker as our task board and did go through the burndown charts to track our pros and cons as teams.

Future Scope

In the future, implementation to track users and views can be made within the CastNXT and EventNXT using the libraries that we have leveraged and an API can be used to display and analyze that data within the CRM. While an attempt was made to integrate this within this development cycle in the CastNXT project, we faced issues with the db integration to the library. This can be expanded beyond these 2 applications as well and other applications can be on boarded onto the CRM so user management of various apps, central reports, email and analytics can sit in one place for all the applications

Project Links

Heroku Deployment:

<https://fashionxt-crm-service.herokuapp.com/>

Pivotal Tracker:

<https://www.pivotaltracker.com/n/projects/2599929>

Source code:

<https://github.com/j-reddy4/FashionNXT-CRM-Service>

Project Demo and Poster Video: <https://youtu.be/kyeKnNjVwlk>