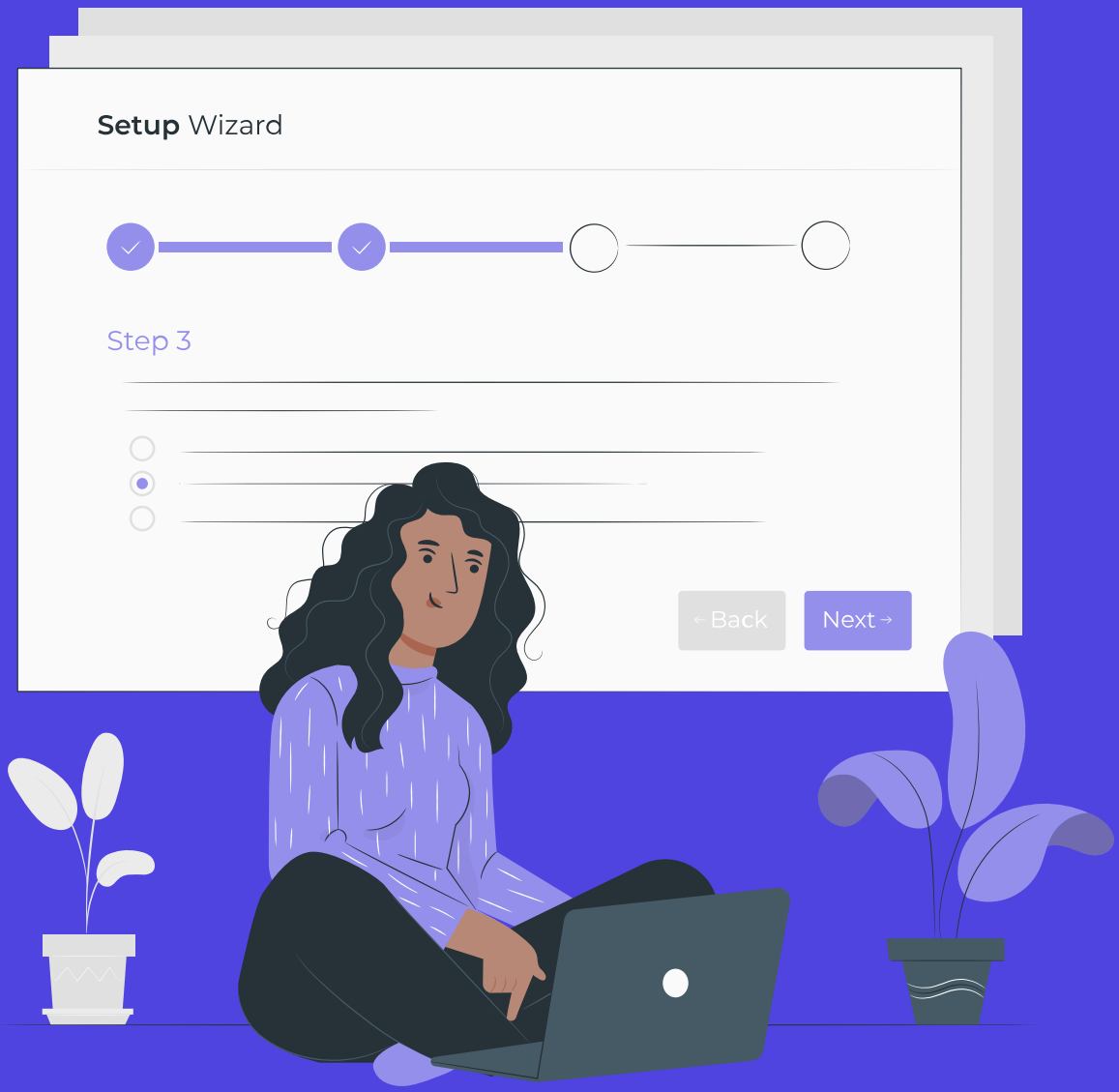
How applications are built in the industry
Lesson Plan



Relevel
by Unacademy

Educator Introduction (5 min)

- Greet the students and highlight the key objectives of the class.

Topics To be Covered In The Class (5 min)

- Scoping
- Use Case diagrams
- Sample scope of an application
- Logging
- Monitoring
- Code reviews
- Architecture design

Building real life applications is not only just about coding features and shipping them to production. In this class we will discuss what are the various steps that take place before and during the development of a product so that it can solve the expected problem for us.

Scoping (10 min)

The most important piece of information that we need to brainstorm, is to take a deep dive into preparation of our Scope Document. Some call it a Scope Statement or a Scope of the given work / task. At its core, it is a fairly simple document that contains as much as possible details of our project. In this document we explain our requirements to our development team. We just need to write the exact specifications in layman's terms.

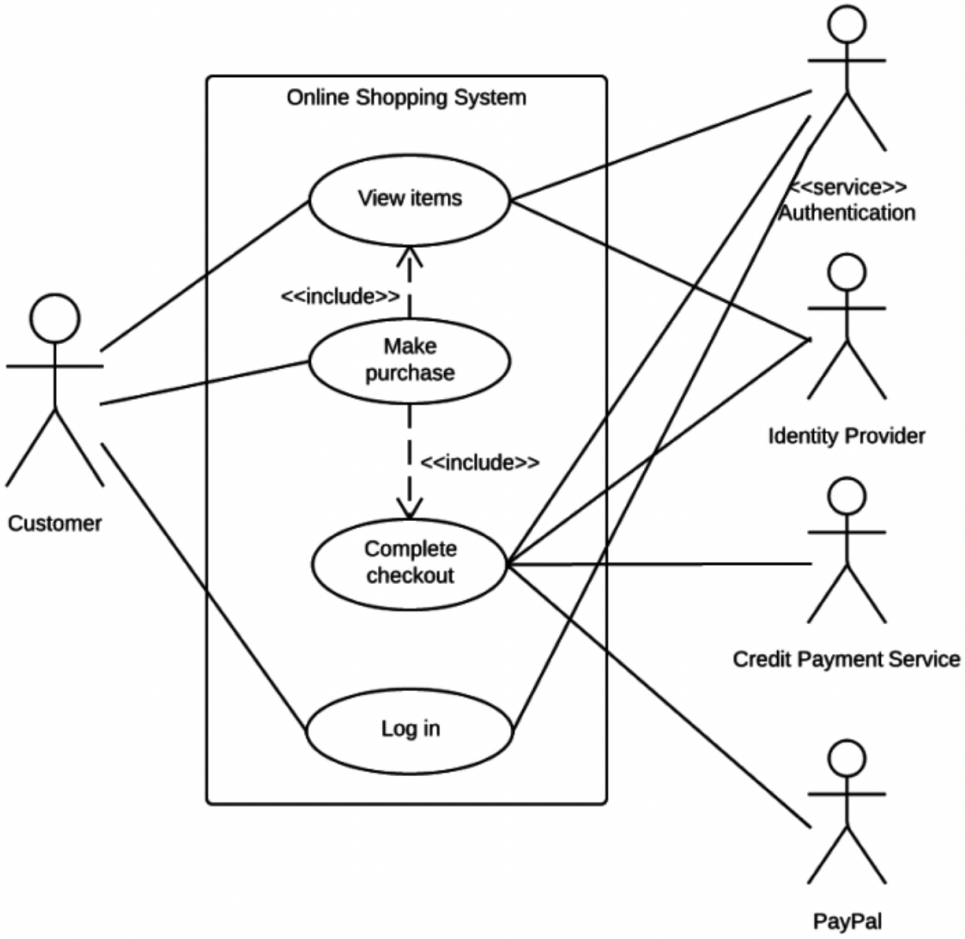
We can simply start mentioning as much detail as we can about everything we want our app to do, how it should behave and work for a user, and any special details that the app will contain. The main agenda for a scope document is to have a document as if we are trying to explain and give clarity to someone what our idea is, who has no idea or prior experience or knowledge on your idea or vision.

All of the minute small ideas add up to the larger vision, capturing as many nooks and crannies as possible that need to be exposed. Try putting yourself in the driver's seat of the app and convey what you see and how it works. The bottom line is that anything at all that you can get on paper is going to be helpful information, and this will give your development company the ingredients to work with to start building out a project quote.

Use case diagram (20 min)

A use case diagram can help us to summarize the details of our application's users (called as actors) and their activities and interactions within the system. To create a use case diagram, we'll take help of specialized symbols as well as connectors. A proper use case diagram will help our team discuss and represent: Situations in which your application interacts with the end users. End Goals that our application helps those entities (called as actors) achieve The scope of our application

Example:



Let's try to write scope document for a food delivery application:
(We can show this scope document to the students in class for demonstration)

Scope Document Sample: (70 min)

Food delivery is fraught with problems both on the consumer as well as the restaurant end. The problems on the consumer end range from reliability and consistency in service, restaurant unavailability and high minimum order values etc. On the restaurant end the overheads pertaining to the maintenance of a fleet of delivery staff, high rentals and marketing costs lend credence to disruption. The driving principle behind HelloEat is to revolutionize the restaurant-takeaway-delivery business.
The product is identified in a mobile APP running on both Android and iOS systems that allows a client to order any restaurant product (for example pizza) from every HelloEat network point (the nearest one or the one that the client wants to select), get promotions and accumulate virtual points in a reward system. It includes a Management system to be putted in every single HelloEat network point via an android tablet app.

The main objective of HelloEat in real-time are :

- To offer a complete solution to the urban foodie in terms of food ordering and delivery from best restaurants located in the vicinity
- To create a single window and include a wide range of food parlours and restaurants under its umbrella
- To revolutionize the restaurant-takeaway-delivery business
- Save time
- Reliability and consistency in service
- To provide the best experience to both our customers & restaurants partners, and improving the system is a key part of providing this experience
- To provide food and delivery ratings being a direct feedback from our customers about their order experience
- Improving the feedback experience for both our customers and restaurant partners, making it more seamless, accurate and purposeful
- To provide customers with timely deliveries and real-time tracking of their order status

1.1 SYSTEM FEATURES

All the features that the app contains are reported in the next point of this document. It is a Multi Language starting with Italian (base) and English (not available at the launch).

1.1.1 Client and home management

This section describes all the functionalities related to the order management system.

Starting the app

Everytime that the app is started an intro is provided that shows simply the logo or a short animation of it .

Select a restaurant / search filter

The user can select automatically the nearest featured restaurant via geolocalization button or insert his own address and select the nearest one. It also shows a preview of 3 most preferred restaurants.

1.1.2 Administration Panel

Main Admin Panel (Android Tablet APP)
They main admin panel must be able to manage clients, orders, fields (categories, products, ingredients, size and other fields), promotions (notifications and/or promo) and access to all the restaurants panel/data. It could create promotions to send via push notifications to selected categories of users (selected by City or other fields to establish in future). It could create promotions that could be shown as push notifications selecting audience too (in this case, it could select all the clients from a restaurant for example). Promotions could be as another form (opening restaurant 3 slide form) with only a simple text and an explication image to insert).
As requirements from the administration, in this panel the Main Admin could check the total of the orders divided by restaurant or the total and we must provide an ID for mobile app orders that is subsequent and ordered only by time (for example, if a client order something from “Restaurant A” at 4 pm and another one order something from “Restaurant B” at 4:00:01, they will be ordered as ID 0000001 and ID 0000002) so not by restaurant but by timing. All the orders could be shown/filtered by restaurant/day/week/year.
It must be possible to calculate and manage a 6% and 2% fees to be considered on all the orders and track it/ generate an invoice for every single HelloEat point every 15 days. It could be a “Fees management” section. The main admin could be able
to change those % with only effective from the moment of the change (not retroactive).

Super Admin Panel (Android Tablet APP)
The super admin panel includes all the functionalities of Main Admin but it can manage only the food points of a chain. He/She cannot touch and see the conditions related to the fees.

Admin Panel (Android Tablet APP)
The admin panel is referred to a single restaurant one. It has all the features of Main Admin but they cannot insert new products but just manage orders and clients.
Every admin could add “local zones” to manage orders. This will be useful to manage deliveries men. They could add few zones to a single delivery man and create their profile (Just a name/nickname) so they can choose to auto assign orders by zone or select them order by order).
Main features are:

- Take an order via call by showing the caller ID that auto identify the user (the tablet that we’ll give to the client has 4g connection and a SIM alleged).
- A panel that makes simple to manage a new order or modify one that is currently on course.
- You can select if the order is a “local one” or if it is a “delivery one”.
- You can easily modify or change the user fields to finalize the order.
- It must manage the receipt that could be double, triple or even more because a single Hellopizza point could have more printer connected (example one in the kitchen and another one in the reception desk and the obligatory fiscal one).
- It must manage different fiscal parameters like VAT (In Italy, it’s 22% but the system could be sold and used in UK so they have less and we must be able to select this thing from the start). Every product price is to be considered as VAT included.

- Non fiscal orders will be hidden in the count of the daily order summary. The admin could unlock them pressing a particular button or pressing a combination of button (keyboard or physical tablet buttons).
- When an order is finalized, it must autoprint the receipt (you can select preferred printers in the admin panel) or select manually the printers. They will be identified with P01-P02-P03 where P01 will be also the fiscal one.
- When a printer is not present in the HelloPizza Point, it must create a pdf with the summary of the order (invoice).
- To finalize a manual order, you can select what type of payment they will make: cash or electronic payment. If you select the second one, it will autogenerate/print the fiscal receipt.

The admin could manage orders delivery time: when a user makes an order from a mobile app/web, the single admin could accept, modify the order time that the user selected (they could choose a range of 15 minutes starting from the HelloPizza single point working time to the closing time). Every change to the original order must be shown to the users via push notifications (order canceled message, or order accepted or delivery time changed to xx:xx).

1.1.3 Categories, products, ingredients, size and fields (registration forms too)

Categories in the order menu

You can find this macro and sub categories in the order menu :

Pizze:

- Pizze Pomodoro
- Pizze bianche

Other categories could be inserted by the Main Admin when a store is configured for the first time (we could arrange an excel or something similar that must be compiled from the client then uploaded from the Main Admin).

Products

It will show all the products like there are exposed on the web-app and they will be given separately.

Main Ingredients

Ingredients are shown under every pizza's name/product divided by a ",". Every ingredient must have a function like + or – to be selected if the client wants more or less of that. If + is added, we must provide a price for that specific thing. (ex. Pizza Margherita with + mozzarella will costs like 0.50 € more. Those kind of pricing must be provided from every new installation).

Removing Ingredients

Base Ingredients are shown under the section "Ingredienti base" as fields to select and they are ingredients that will be removed from the main ones.

Extra Ingredients

Extra Ingredients are shown under the section "Ingredienti Extra" as fields to select and they are ingredients that will be added from the main ones. Those ingredients will be provided by the admin via compiling the excel document.

User registration fields

These are the fields that a user must compile to register and finalize an order:

- Name
- Surname
- E-mail
- Phone (confirmation via sms)
- Address
- Number of address
- Password

All the user registration field are equal to the ones on Web-App software

Order withdraw menu

You can select if withdraw the order directly in the restaurant previously selected or if to make it delivery to home and those are the relative category and fields:

- Preferred withdraw/delivery time with this fields (Day of withdraw with fields for 3 day: Today, Tomorrow and the day after with the correct name of the day)
- Comment (this must be a fillable field that will delivery additional info on order or the delivery)

Payment menu

You can select if pay with cash at the delivery man/restaurant or via Cards through PayPal.

“Run” Order

The system must provide an option called “Run order” that the user could select in the order withdraw menu phase of it. It has on overprice of € 5.90 and every restaurant must provide how much “run user” they can serve in 1 hour. It is a feature that makes the order become a priority in the selected delivery time so it will become the first to be prepared and it must has something different in the UI (in the tablet app management).

1.2 OTHER DESCRIPTION

1.2.1 Product Perspective

These mobile Apps are going to extend the Hellopizza offer replacing their use of the JustEat system and bring a brand new management software that will substitute their internal management.

1.2.2 Product Features

Product features of the app includes a geolocalization/selection of the nearest HelloEat network point, all the things related to order a pizza, a Facebook interaction on sharing ordered pizzas, a reward system, an overall counter, promotionals interface, paypal payment system and eventually another gateway (stripe or something similar with lower fees), one click order.

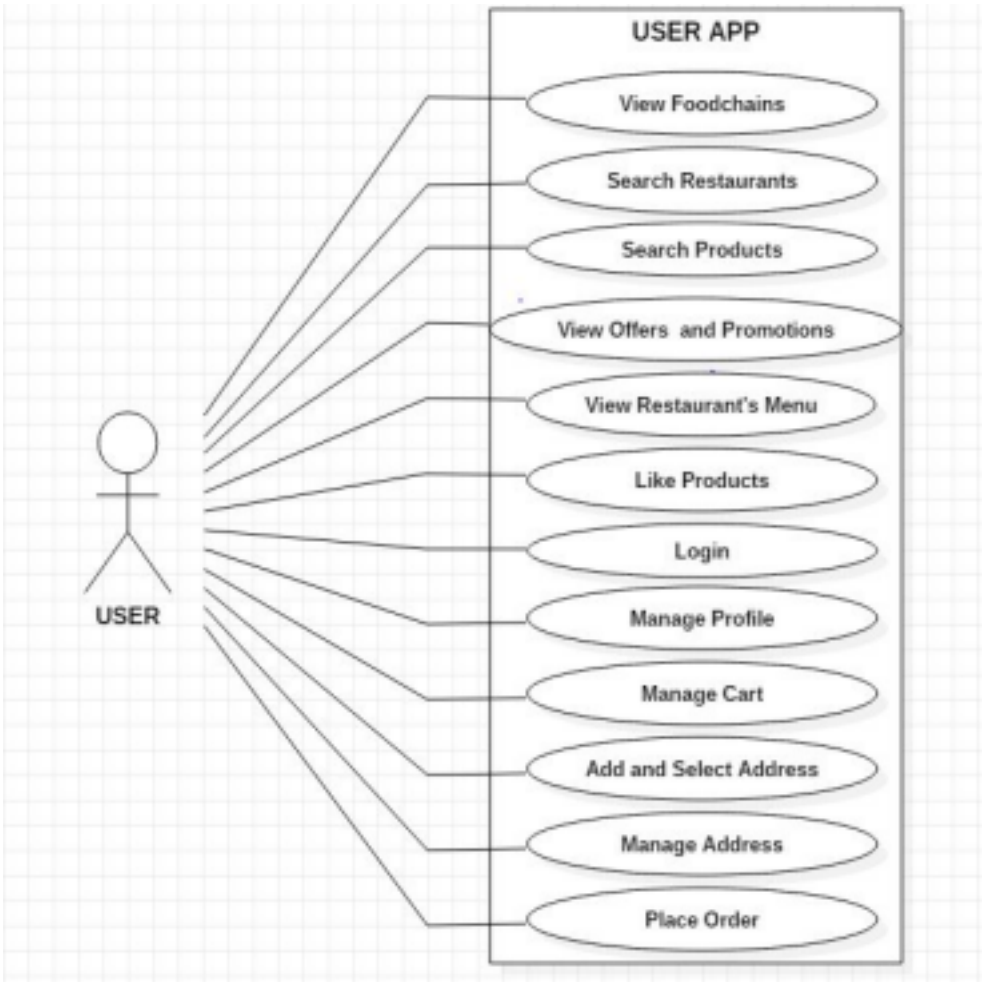
1.2.3 User Classes And Characteristics

There are 5 types of users in this application. They are:

- Clients ← clients are basically every single user that will utilize the app.
- Pizzaboy ← delivery pizza man that could report a successful or not delivery.
- Admins ← an admin is the manager of a single restaurant.
- Super Admin ← the owner of a food chain – he can manage the food chain points over the admins.
- Main Admin ← the owner of HelloEat Company / CEO – it must has control on the admins too.

2.1 USE CASE DIAGRAM

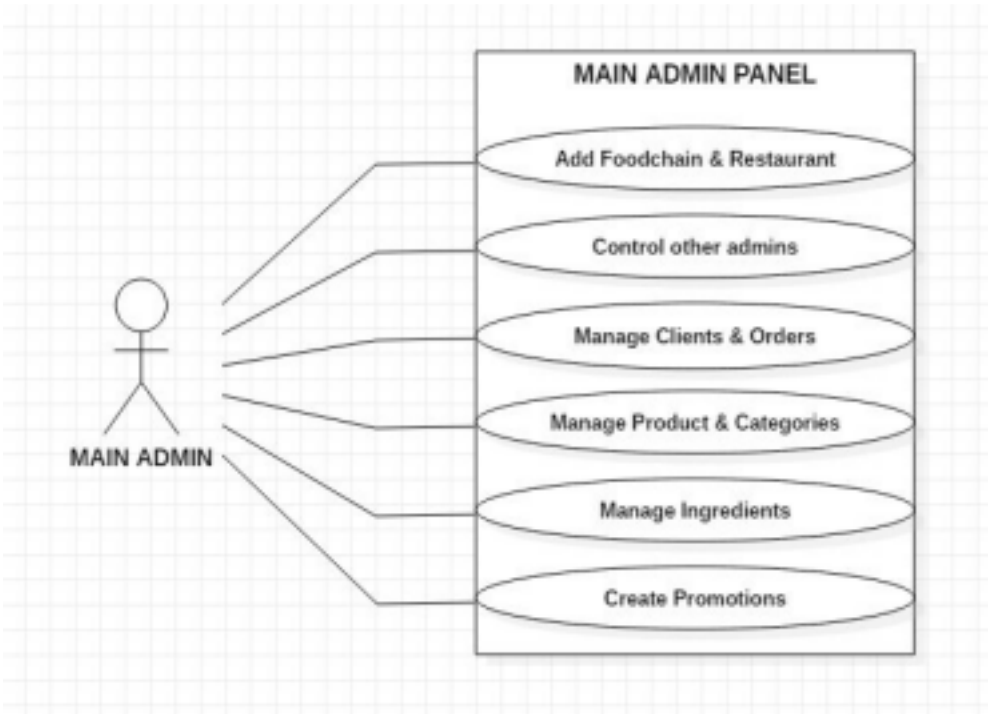
2.1.1 User App



Use Case Description

Clients are basically every single user that will utilize the app. A user can view foochains in the home page of the application. A user can also search restaurants and products in the explore page, view offers and promotions, view a particular restaurant's menu, like specific products, login to his/her account, manage profile, manage cart, add and select address, manage address i.e. edit and delete his/her address. A user can also place an order.

2.1.2 Main Admin Panel

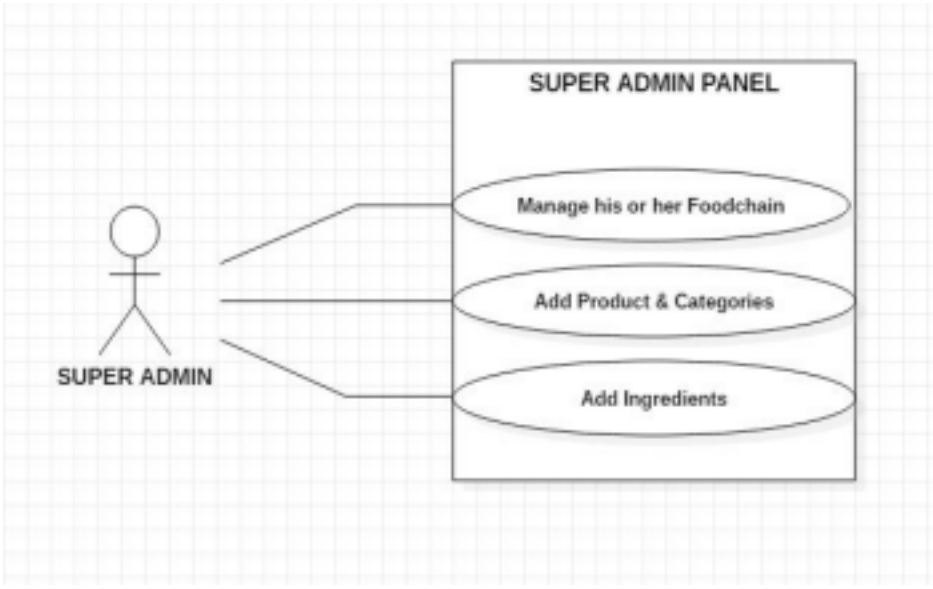


Main Admin Panel

Use Case Description

The Main Admin is the owner of HelloEat Company / CEO – it must has control on the admins too. They main admin panel must be able to manage clients, orders, fields (categories,products, ingredients, size and other fields), promotions (notifications and/or promo) and access to all the restaurants panel/data. It could create promotions to send via push notifications to selected categories of users
(selected by City or other fields to establish in future). It could create promotions that could be shown as push notifications selecting audience too.

6.1.3 Super Admin Panel

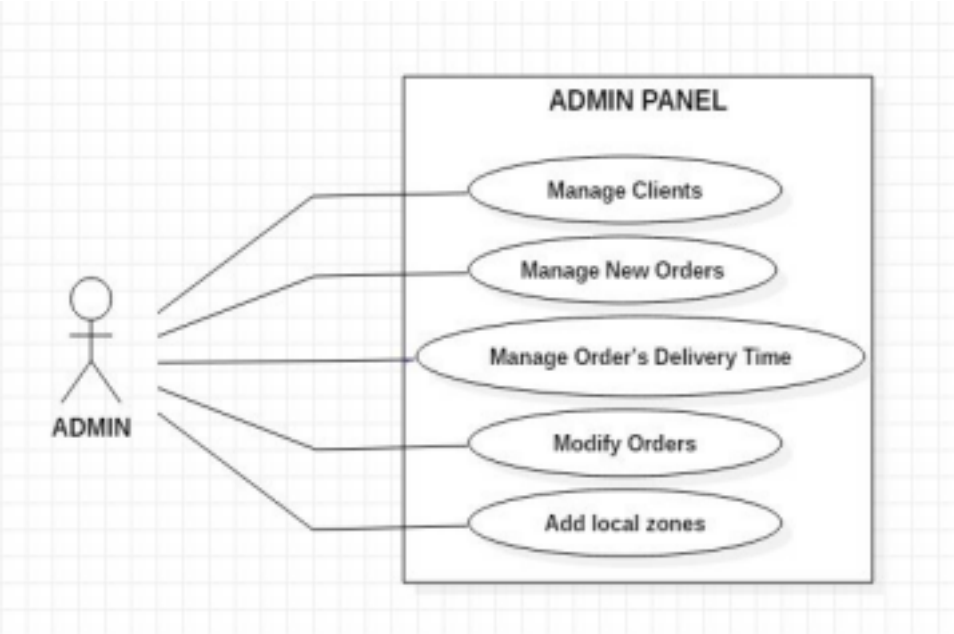


Super Admin Panel

Use Case Description

A Super Admin is the owner of a foodchain – he can manage the foodchain over the admins.The super admin panel includes all the functionalities of Main Admin but it can manage only the food points of a chain. He/She cannot touch and see the conditions related to the fees.

2.1.4 Admin Panel



Admin Panel

Use Case Description

An Admin is the manager of a single restaurant.The admin panel is referred to a single restaurant one. It has all the features of Main Admin but they cannot insert new products but just manage orders and clients. Every admin could add “local zones” to manage orders. This will be useful to manage deliveries. They could add a few zones to a single delivery man and create their profile (Just a name/nickname) so they can choose to auto assign orders by zone or select them order by order). An admin also manages new orders and manages order’s delivery time.

Logging And Monitoring (20 min)

Login is a practice of logging errors and changes or logs of applications that are collected over time. The purpose of the login is to create a continuous record of events that take place in the operating system. Log files can help us to update any event within the application, errors, failures. Along with that login messages can give useful information to identify the cause of the problem. Log data can help us to solve problems by identifying which changes lead to the error, but it is as important as the information it contains.
It also serves other purposes, such as making written records for inspection and compliance purposes, identifying trends over time, and obtaining sensitive information. Logging plays an important role in the use of all sizes, but it should be used with caution. Avoid storing, forwarding, or checking external information by prioritizing activities. Entering too much data can create a hassle of resources, both in terms of cost and time. A good logging strategy usually provides two types of data: systematic data and a system that notifies system administrators of a potential problem.

An umbrella word monitoring can include many aspects of system testing, but in this context, we are referring to application performance monitoring (APM). APM is an application process that collects, compiles and analyzes metrics to better evaluate system usage by measuring availability, response time, memory usage, bandwidth, and CPU usage.
Monitoring systems rely on metrics to alert IT teams to confusing performance across all applications and cloud services. Ideally, teams will use tools and monitoring across systems.

Logging and monitoring are two different processes that work together to give a wide range of data points that track the health and performance of our infrastructure. APM uses in-app metrics to measure availability and manage performance. Login creates a log of events generated from applications, devices, or web servers that serve as a detailed record of activity within the system.
Using a combination of log management to collect, organize, and update data and monitoring tools to track metrics provides a comprehensive view of your system's availability and detailed information on any issues that may affect user information. APM tells you how apps behave and logs data from apps, network infrastructure, and web servers provides a great insight that will tell you why the app works as it is. An effective logging strategy improves app performance monitoring.
Figuratively speaking, monitoring metrics are like a security alarm that alerts potential intruders; log files serve as a security camera image that will give you clues to tell you what happened and how. There will be some conditions to use when you will need only one or the other, but having both gives you great ability to fully understand your system and its vulnerability.
Ultimately, your goal is to keep the apps healthy and user-friendly. By combining logging and monitoring to achieve this goal, your developers and working groups will be able to plan and solve application problems quickly. Create an effective strategy to improve the integration of your logging and management through the following key processes:

Enable both methods to work together
If your main goal is to improve the benefits of analyzing log file data and application metrics, simplify that process by configuring your system to send log data directly to your monitoring tool. Saving file log files to the disk or sending them to the cutting tool only creates a lot of drag on resources and a potential workflow bottle. Make sure your monitoring tool supports the programming language of your application to ensure compliance and ease of use.

Log the right data
Log data needs to tell a short but complete story. Data should select, define, and provide the right context to help solve the problem. Useful log data usually includes things that can be done, and includes information such as timestamp, user IDs, time IDs, and metrics for application usage. Collecting the full range of active data enhances the information obtained from your monitoring tool.

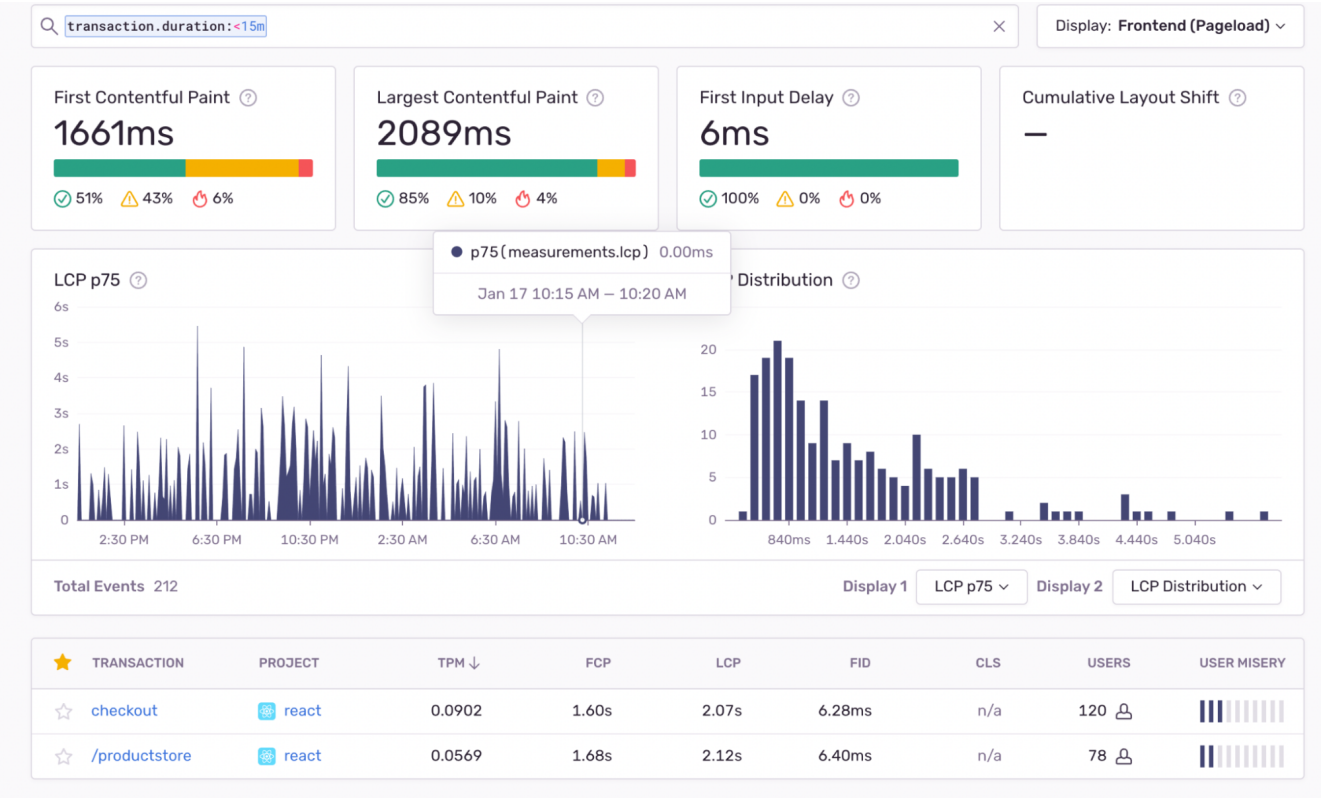
Use structured log data
Organize your data by making it easy to search, identify, and store to make sure it's organized. Structured data provides a complete overview of what happened, and can provide your monitoring tool with unique identifiers such as which customer ID has encountered an error. Providing customer ID information obtained by login allows your monitoring tool to identify how that particular user was affected, as well as what other problems you may be experiencing as a result.

Example Logs:

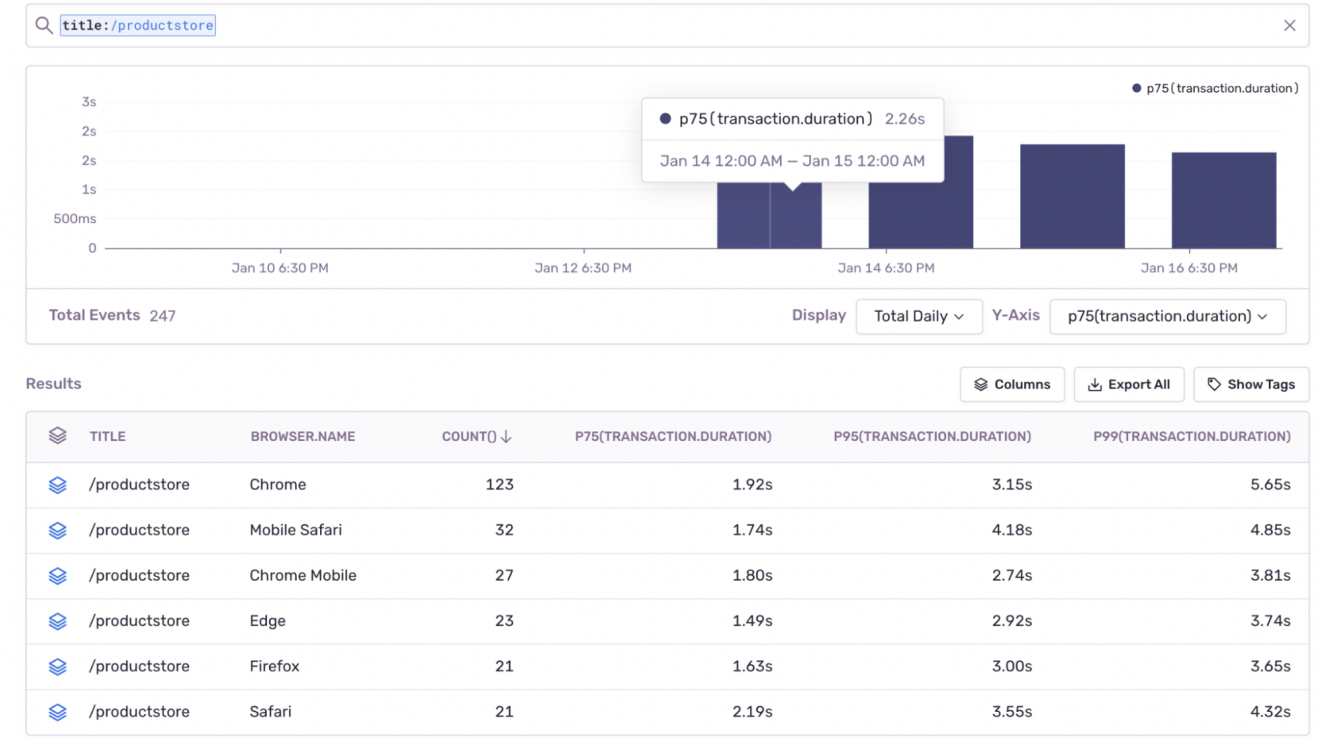
```
 01
03/22 08:51:01 INFO   :.main: *************** RSVP Agent started ***************
 02
03/22 08:51:01 INFO   :...locate_configFile: Specified configuration file: /u/user10/rsvpd1.co
03/22 08:51:01 INFO   :.main: Using log level 511
03/22 08:51:01 INFO   :..settcpimage: Get TCP images rc - EDC8112I Operation not supported on
 03
03/22 08:51:01 INFO   :..settcpimage: Associate with TCP/IP image name = TCPCS
03/22 08:51:02 INFO   :..reg_process: registering process with the system
03/22 08:51:02 INFO   :..reg_process: attempt OS/390 registration
03/22 08:51:02 INFO   :..reg_process: return from registration rc=0
 04
03/22 08:51:06 TRACE  :...read_physical_netif: Home list entries returned = 7
03/22 08:51:06 INFO   :...read_physical_netif: index #0, interface VLINK1 has address 129.1.1.
03/22 08:51:06 INFO   :...read_physical_netif: index #1, interface TR1 has address 9.37.65.139
03/22 08:51:06 INFO   :...read_physical_netif: index #2, interface LINK11 has address 9.67.100
03/22 08:51:06 INFO   :...read_physical_netif: index #3, interface LINK12 has address 9.67.101
03/22 08:51:06 INFO   :...read_physical_netif: index #4, interface CTCD0 has address 9.67.116.
03/22 08:51:06 INFO   :...read_physical_netif: index #5, interface CTCD2 has address 9.67.117.
03/22 08:51:06 INFO   :...read_physical_netif: index #6, interface LOOPBACK has address 127.0.
03/22 08:51:06 INFO   :....mailslot_create: creating mailslot for timer
03/22 08:51:06 INFO   :...mailbox_register: mailbox allocated for timer
 05
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP via UDP
 06
03/22 08:51:06 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address n
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 129.1.1.1, entity for rsvp allocated and
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address n
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.37.65.139, entity for rsvp allocated a
initialized
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address n
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.100.1, entity for rsvp allocated an
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address n
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.101.1, entity for rsvp allocated an
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.116.98, entity for rsvp allocated a
initialized
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO   :.....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.117.98, entity for rsvp allocated a
initialized
```

Please show this link to the students for demonstration of monitoring and explain

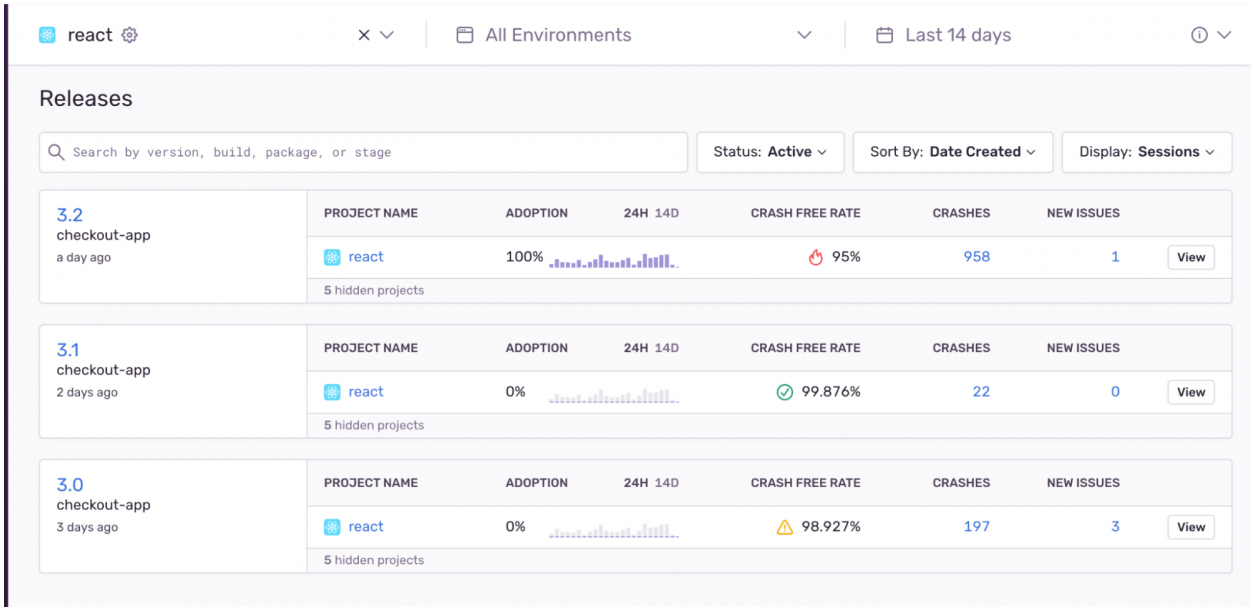
different aspects of the dashboard



Created by: – | Last edited: an hour ago



Performance metrics



Release schedule

Code Reviews (20 min)

In simple terms, code review is the process by which a developer code (or pull request) is tested by a peer or a senior engineer. Used properly, code updates help developers to identify common bugs quickly and reduce the amount of work required to upgrade the code in later stages.
Code review is an important part of DevOps cycles, as it helps to ensure that the code is as clean as possible in the first step of the development pipeline. In fact, it is one of the tests that every engineer should do before pushing his code into a major branch.

Code reviewers primarily verify the following parameters:

- Is the code well designed?
- Did the developer use readable names for classes, variables, and methods for ease of understanding?
- Comments – Are the comments meaningful and clear?
- Can the code be made simpler? Will it be easy for future developers to understand the code?
- Does the code include required and correctly crafted automated tests?

One might think of code review as just another routine check for identifying common bugs. However, that's not true. Code review has many more benefits to offer than just the identification of bugs. Listed below are key benefits achieved using the code review process.

1. Ensures consistency in design and implementation
Every developer has a unique coding style. Needless to say, many engineers are involved in major projects. When engineers continue to follow their different writing styles during development, it disrupts collaboration and hinders overall progress.
The code review process compels developers to follow specific coding processes throughout the sprint development phase. This method balances the source code, making it easier for all engineers (even newer ones) to read and understand.

Code updates are helpful over time as team members continue to make changes to projects. Additionally, maintaining a consistent writing pattern will help future engineers spend more time creating new features rather than wasting time analyzing existing code.

2. Optimizing code for better performance

Code review helps young developers identify areas where they can improve when it comes to using codes. Since young engineers do not have the necessary knowledge, they do not recognize certain coding techniques that can help them write pure code. The code review process helps them find the right answer for senior engineers and as a result helps them hone their coding skills. In addition, it also helps to identify serious flaws or mistakes that can lead to serious fetuses.

Because of the boring nature of systems, even the most experienced engineers will inevitably ignore mistakes. Code review helps eliminate these errors before moving on to the next steps by inviting new pairs to update each unit of code. The reviewer then checks and removes errors, if any.

3. Collaborating and share new techniques

In addition to saving time and money, code updates also deliver a personalized ROI. The programmers spend most of their time alone writing codes. However, practices such as code revision force developer interaction. This encourages developers to work together on their code and to exchange ideas. In addition, it also promotes trust among developers.

Alternatively, team members can exchange information about new readings during code review. This helps team members improve their skills by acquiring the latest technical knowledge.

4. Monitoring project quality and requirements

Every project has a variety of well-defined requirements and needs. In addition, a few engineers are involved in building a significantly complex project. Each software engineer creates different features depending on the given requirements. However, there is a good chance that an engineer may interpret the text unnecessarily, and the person may end up developing an insignificant feature. Code review helps to deal with such complex situations as the process confirms that after a few iterations there will be an improved feature against the expected feature.While code updates may sound like another standard test, teams gain more than bug fixes. Improved collaboration, improved learning, improved code validation, and systematic improvements are important benefits of code updates. In addition, it also serves as a guide for new developers to hone their skills to become professionals. Given this, one can decide how code review adds value to the development cycle of delivering quality products.

Tools for code reviews:

- Github
- Gitlab
- Bitbucket

Example Code reviews and comments

peter-hank reviewed on 17 Jun 2019 — View changes

`client/styles/global.scss` Outdated — Hide resolved

```
154 + }
155 +
156 + #resetpasswordsubmitbutton {
157 +   width: 40%;
```

peter-hank on 17 Jun 2019 — Contributor

indentation

singhsanket143 on 19 Jun 2019 — Contributor, Author

@peter-hank I have made some changes for indentation. Please review them and if there are any more suggestions then do let me know

peter-hank on 2 Jul 2019 — Contributor

@singhsanket143 can you use some online tool like http://www.calluna-software.com/Formatter/Css and fix the whole file?

singhsanket143 on 4 Jul 2019 — Contributor, Author

@peter-hank Updated the indentation

peter-hank on 5 Jul 2019 — Contributor

@singhsanket143 it's still not nice, can you make sure the whole file looks good?

singhsanket143 on 7 Jul 2019 — Contributor, Author

@peter-hank I have used https://www.freeformatter.com/css-beautifier.html as a tool to indent the whole file. Please review it.

Reply...

Unresolve conversation — **peter-hank** marked this conversation as resolved.

peter-hank reviewed on 17 Jun 2019 — View changes

`client/views/resetPassword/resetPassword.js` Outdated — Hide resolved

```
10 + Template.ResetPassword.events({
11 +   'click #resetpasswordsubmitbutton': function(event, template) {
12 +     event.preventDefault();
13 +     var password = $("#resetPassword").val(); // Get the new password
```

peter-hank on 17 Jun 2019 — Contributor

comments above or below the code please

peter-hank on 17 Jun 2019 — Contributor

single quotes as well

singhsanket143 on 19 Jun 2019 — Contributor, Author

@peter-hank Updated the quotes style

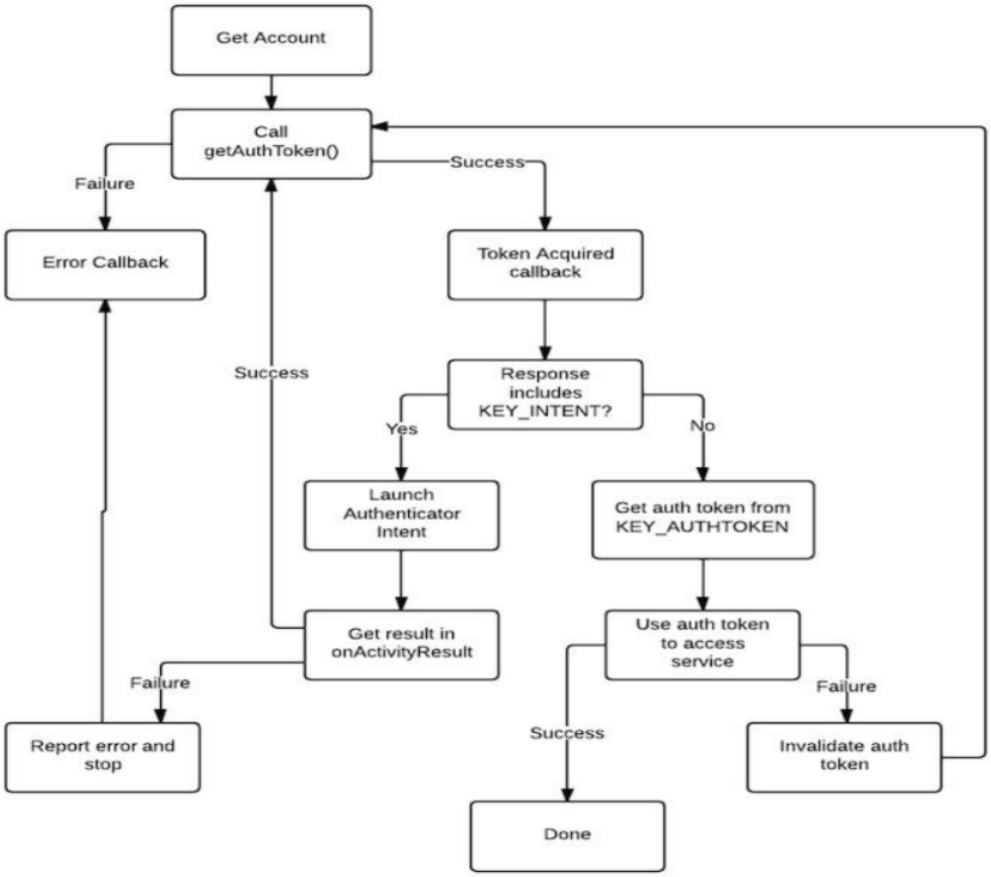
Reply...

Unresolve conversation — **peter-hank** marked this conversation as resolved.

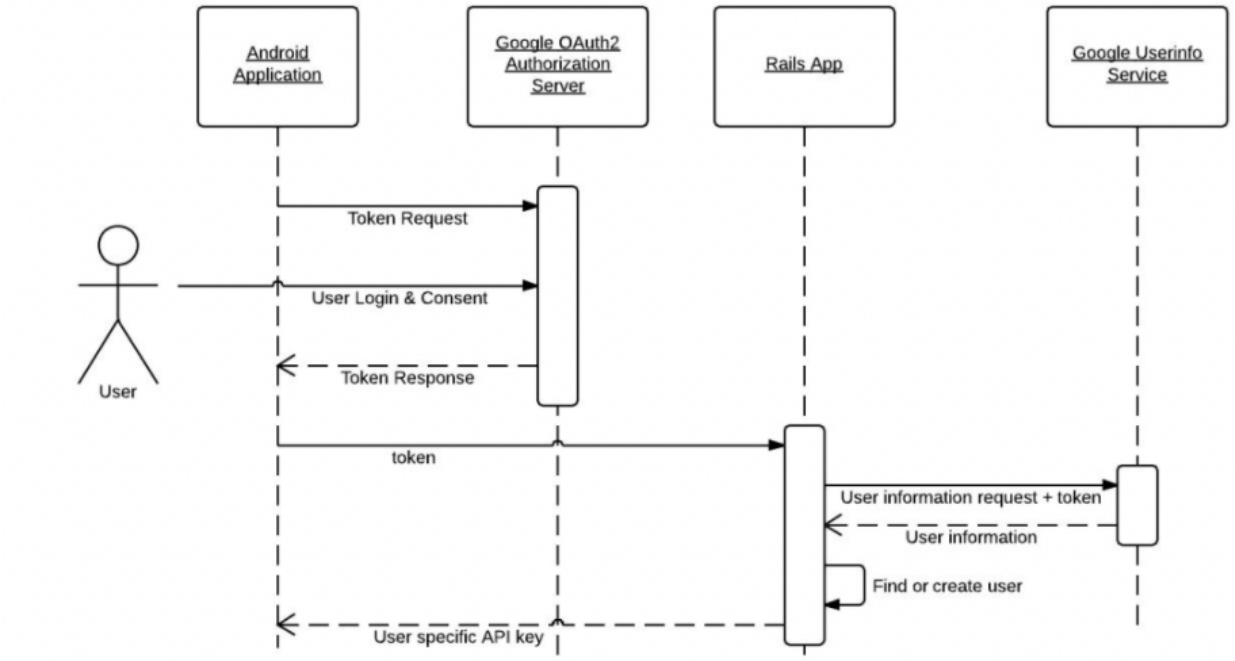
Architecture Design (20 min)

Each Section of our app is expected to be designed in terms of architecture. The architecture design of a module explains about how the whole lifecycle of the module is going to be handled, which services are involved, which kind of data they are going to expect and what will be the api signatures, what will the database looks like, along with estimation about how many queries are going to be handled by the service, what measures we have taken to make it more scalable.

Below is the architecture design for a google/fb login module.

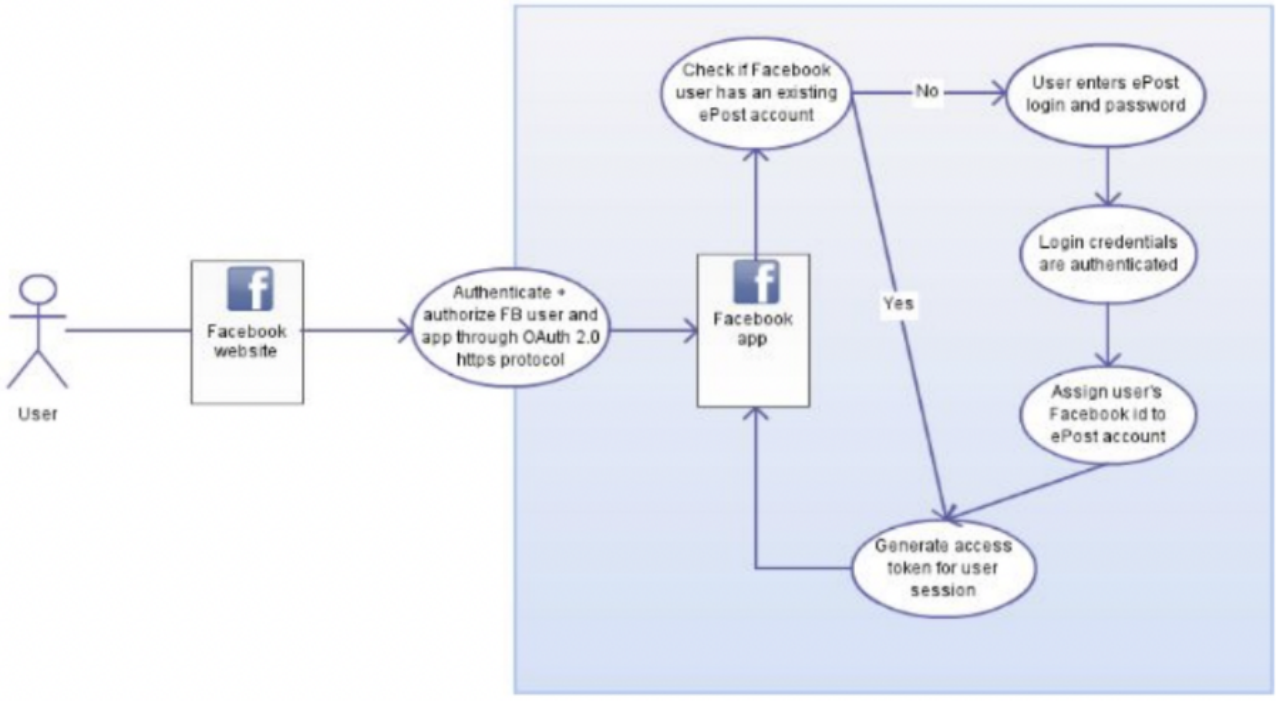


Data flow diagram (Explain every component separately)



Sequence diagram

The above module can be a part of our HELLO EAT application that we discussed. In the above architecture if we see the sequence diagram it demonstrates how step by step the data is moved in the system to get the desired result. Here we can see that the user first raises a request to fetch an authentication token from google omni auth servers and in response it gets the unique auth token.
Then whenever a user needs to call any service it first authenticates itself with the help of a token, the token is validated and if it successfully validates then we get access to the server resource and then we are able to use the service accordingly.



Use Case Name	Login to the application
Brief Description	The login process includes signing up users using google and facebook accounts.
Flow Of Events	1. User should click the sign in with google or sign in with facebook button. 2. The account selection dialog box appears where user can choose the respective account. 3. Authorization of the selected account. 4. Storing the important details for the user in the database and generating an API key for them.
Actors	User and Google And Facebook Auth API
Pre Conditions	A user must have a google or facebook account.
Post Conditions	User will be able to use rest of the features of the app. If the user is already logged in somewhere then he/she will not be able to use the features on the earlier login and will be logged out.
Exceptional Conditions	If the user enters wrong login details then the login process is cancelled and the user is notified.