MSIS 2606 Software Project Management

Final Project Report

Team 4 - Aashka Aradhya | Aishwarya Dingre | Megha Prasad | Megha Sharma | Ritika Mathur | Oleksandr Vozniuk | Yue Wu

Contents

Contents

Style Up's Mission Statement

It is our mission to bring our clients the clothes they love by leveraging our personal stylists and combining their expertise with technology, to deliver an affordable styling solution, which adapts to our client's evolving preferences.[1]

1. Mission Statement: https://www.stitchfix.com/about (Slide 3)

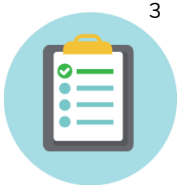
About StyleUp

We are a team of 75 and growing as our customer base grew 240% from 500 to 1200 in 10 months

What does StyleUp do?

At StyleUp, we are transforming the way people find what they love. Our clients want perfect clothes as per their preferences without the burden or search or having to keep up with current trends


2


3

Fill out a style profile

Custom selected clothing delivered to your door


4

Buy what you like, send back the rest. Shipping is free and easy both ways


5

2. Confused Shopper Image: https://bit.ly/2TVZ4gW (Slide 5)
3. Survey Image: https://bit.ly/2TWxJv1 (Slide 5)
4. Delivery Person Image: https://bit.ly/2BH3gKa (Slide 5)
5. Happy Shopper Image: https://bit.ly/2EgRecs (Slide 5)

How do we do it?

6

Our merchandise is curated from the market, ensuring something for everyone

Rich data on both sides of the market, makes StyleUp a matchmaker, connecting clients with styles they love and never would've found on their own

Each client fills out a profile upon signup that's calibrated to get us the most useful data with the least client effort to capture dimensions and style preferences

7

6. Brands Image: https://bit.ly/2GvsjUT (Slide 6)
7. Diverse Shoppers Image: https://bit.ly/2DUcx1S (Slide 6)

Problem Statement: Data we need is changing as we grow

Traditionally we curated styles for our customers based on **structured data** collected from the style profile they filled out on sign up

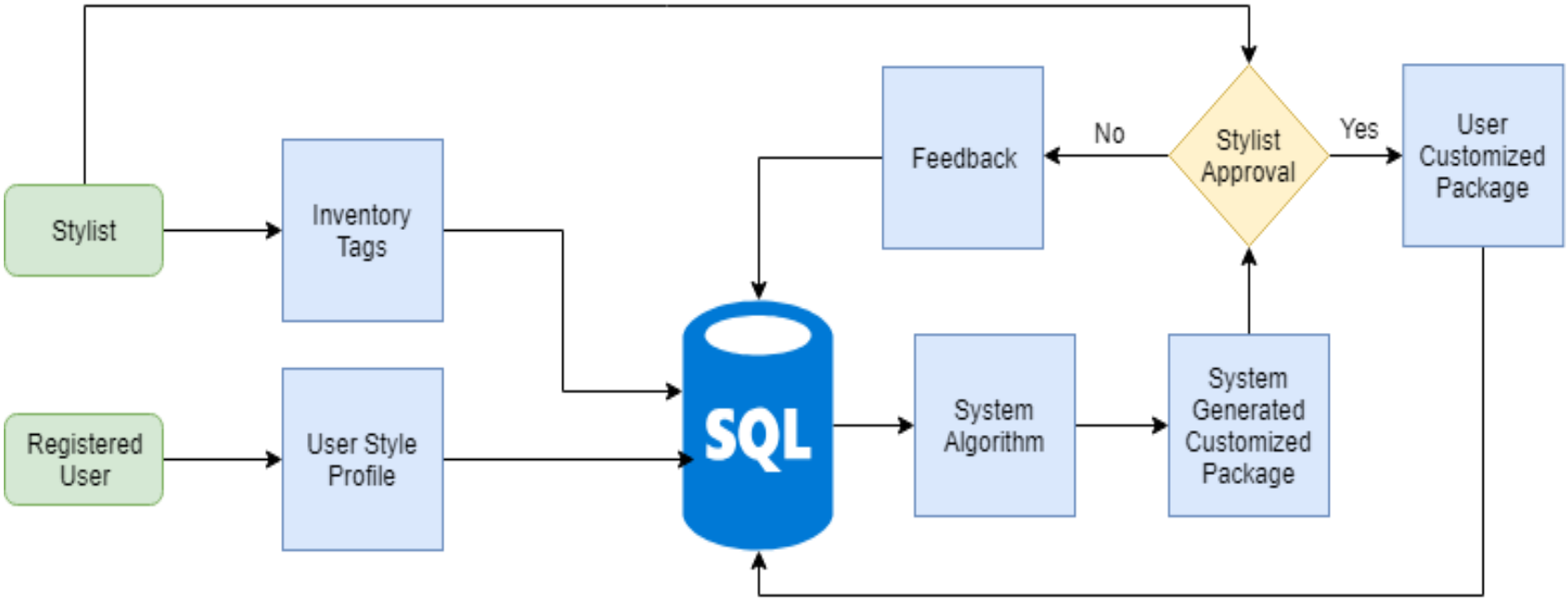
[8]

We plan to explore a lot of photographic and textual data (**unstructured**): inventory style photos, Pinterest boards, social media profiles and the vast amount of written feedback and request notes and photographs we receive from clients

To sustain our growth, we have to enhance our value proposition :
Understand to our best ability what our customers like

8. Growth Image: https://bit.ly/2towo4C (Slide 7)

Our Existing System[9]



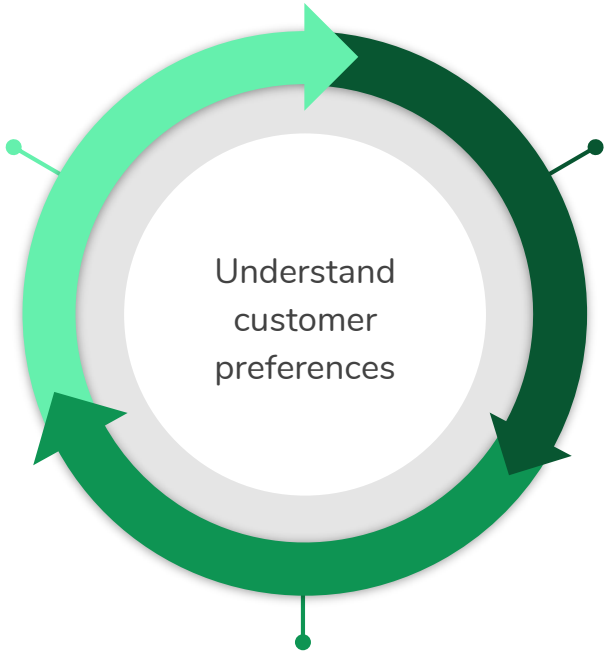
9. Existing System Flowchart drawn using: https://www.draw.io/ (Slide 8)

Challenges we face with MySQL

Unstructured Data	Schema Inflexibility	Scalability	Oracle owned, not community-driven
We plan to capture **unstructured** data from various sources which is not cost-efficient to store in MySQL [10]	We are constantly enhancing our data sources to understand our client's preferences which makes schema flexibility and adaptability indispensable but MySQL has a rigid schema structure	MySQL was originally designed as a single-node system and not with the modern data center concept in mind. sharding solutions in MySQL are manual and make application code more complex. Any performance gain is lost when queries must access data across multiple shards [12]	Lack of community makes MySQL developers inaccessible which makes it challenging to modify or customize enhancements which is against the spirit of Open Source [11]

10. Overwhelmed Person Image: https://bit.ly/2TVjN4B (Slide 9)
11. MySQL Oracle owned, not community driven: https://smartbear.com/blog/test-and-monitor/5-reasons-its-time-to-ditch-mysql/ (Slide 9)
12. MySQL Scalability issue: https://www.gridgain.com/resources/blog/5-limitations-mysql-big-data (Slide 9)

Business Requirements



Create a 360 degree style profile of each customer by combining new and existing data

Create a chatbot module to manage customer feedback and use the data collected to enhance their 360 degree profile

Real-time metrics to handle inventory management and brand collaborations effectively

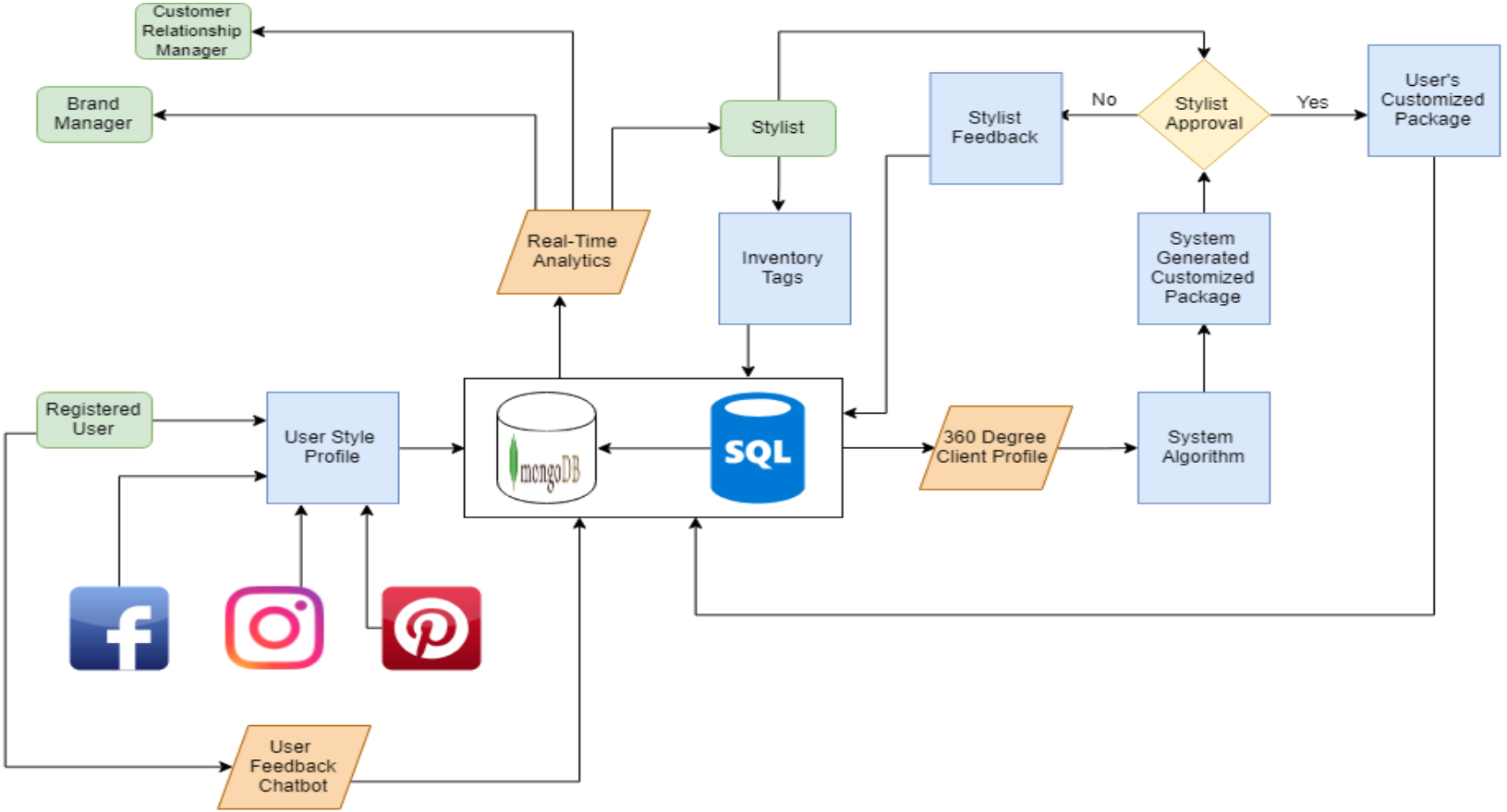
Proposed Solution:

mongoDB[13]

Flexibility of schema	Scalability (Auto-Sharding)	Document-based storage	Easy to learn	MongoDB Charts
Since our data is dynamic, MongoDB would be perfect	In terms of Cluster, Performance and Data	It pairs each key with a complex data structure	Easy to set up and manage	Visualize live data from any MongoDB instance
It can accommodate our needs to store both structured and unstructured data	As we grow, our database could be distributed across nodes, could sustain 100,000 + database read/writes per second and store 1 billion + documents	This would make it easier to develop 360 degree style profile of customers and also link customer chat history to their profile	Given our small team size, MongoDB would be a good choice	Its real-time visualization features help our team to monitor essential metrics efficiently

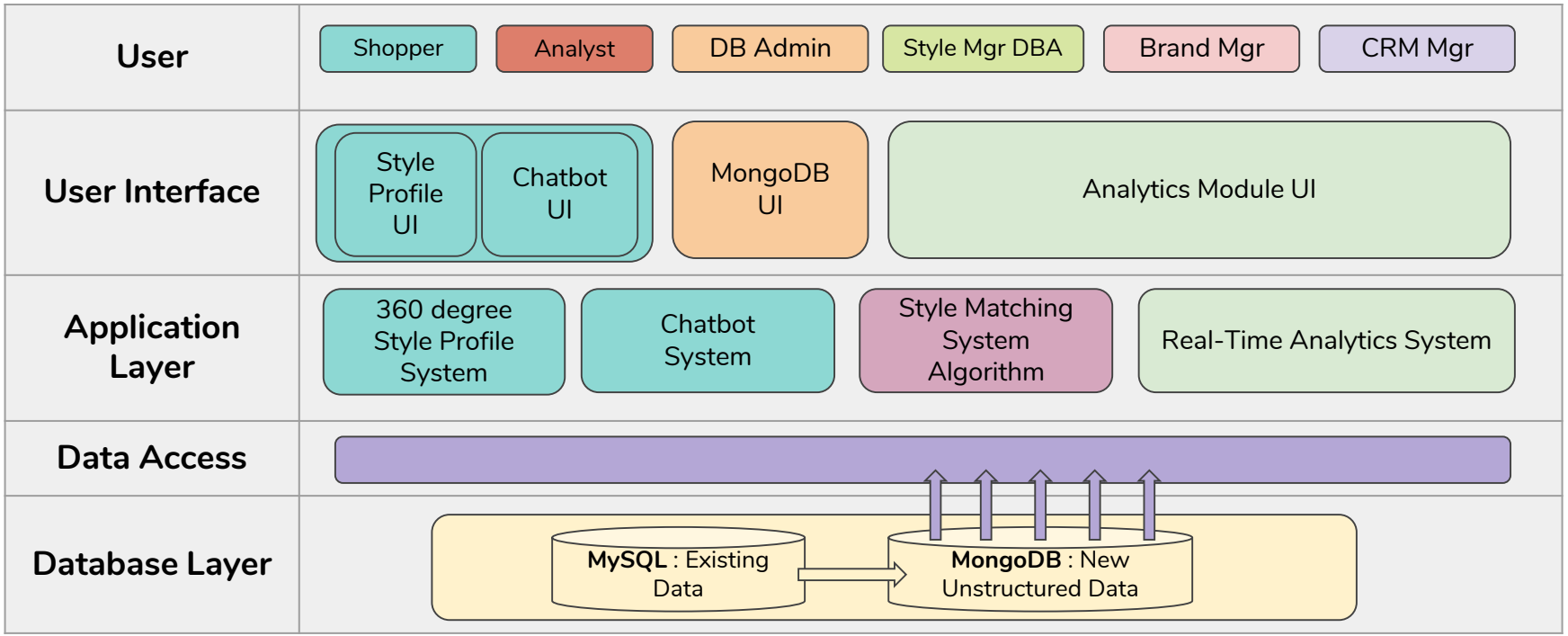
13. Features of MongoDB: https://docs.mongodb.com/manual/introduction/ (Slide 11)

Our Proposed System[14]



14. Proposed System Flow Chart drawn using: https://www.draw.io/ (Slide 12)

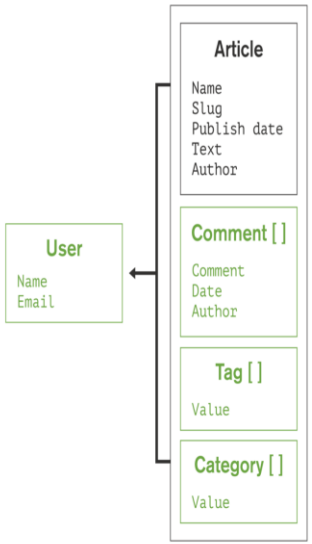
Architecture



Why MongoDB?

Why MongoDB suits us better?

1. **Developer productivity is increased with JSON Documents.**[15]



[16]

Working with data as flexible JSON documents, rather than as rigid rows and columns, is proven to help developers move faster. With data for an entity stored in a single document, rather than spread across multiple relational tables, the database only needs to read and write to a single place. Having all the data for an object in one place also makes it easier for developers to understand and optimize query performance.

This feature would be critical to store our client's unstructured data and create 360 degree client profiles, which would be enhanced by our prospective chatbot feature, as mondoDB has the capability to store Chatbot conversations with the client as documents linked to their 360 degree client profile

15. Developer productivity is increased with JSON Documents: https://www.mongodb.com/compare/mongodb-mysql (Slide 15)
16. Document storage Image: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 15)

Why MongoDB suits us better?

2. MySQL's JSON data type does not bring the developer productivity benefits of a document database to MySQL:[17]

- **Proprietary Extensions:** Querying and manipulating the contents of a JSON document requires the use of separate MySQL-specific SQL functions to access values, which is not familiar to our developers. **The MongoDB API is preferred by our team as it is adopted by industry standard tools and connectors**

- **Legacy Relational Overhead:** Even with JSON support, MySQL users are still tied to multiple layers of SQL/relational functionality to interact with JSON data – low level JDBC/ODBC drivers and Object Relational Mappers (ORMs). **MongoDB drivers are implemented in the methods and functions that are idiomatic and natural to the programming languages used by developers**

17. MySQL's JSON data type does not bring the developer productivity: https://www.mongodb.com/compare/mongodb-mysql (Slide 16)

Why MongoDB suits us better?

2. MySQL's JSON data type does not bring the developer productivity benefits of a document database to MySQL:[18]

- **Complex Data Handling:** When using JSON data, MySQL drivers do not have the capability to properly and precisely convert JSON into a useful native data type used by the application. This includes different types of numeric values (e.g. floating points, 64-bit integers, decimals) timestamps, and dates, or a Map or List in Java or a Dictionary or List in Python. **Binary Encoded JSON (BSON) used by MongoDB and its drivers supports advanced data types not supported by regular text-based JSON**

- **No Data Governance:** MySQL offers no native mechanism to validate the schema of JSON inserted or updated in the database, so developers need to add either application or database-side functionality to apply governance controls against the data. **Schema validation, based on the JSON Schema IETF standard, allows developers and DBAs to define and enforce a prescribed schema structure for each MongoDB collection. This feature would help streamline data collected from our Chatbot module.**

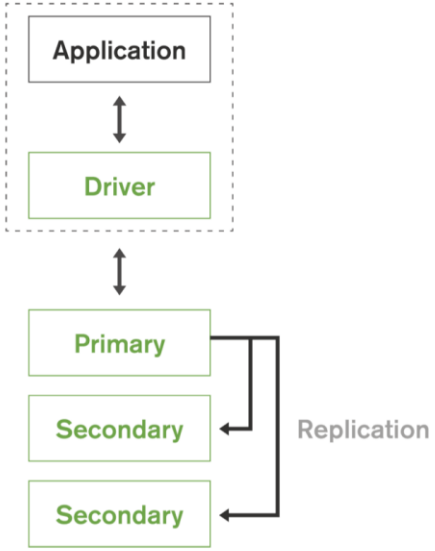
18. MySQL's JSON data type does not bring the developer productivity: https://www.mongodb.com/compare/mongodb-mysql (Slide 17)

Why MongoDB suits us better?

19

3. MongoDB maintains multiple copies of data called replica sets using native replication:

20



A replica set is a fully self-healing shard that helps prevent database downtime and can be used to scale read operations. Replica failover is fully automated, eliminating the need for administrators to intervene manually.

This feature would be critical as we expect our clients to interact with our Chatbot at any time of the day and we have to ensure our Chatbot has access to client's information at all times to provide a good experience and to collect their feedback coherently.

19. MongoDB maintains multiple copies of data: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 18)
20. Replication Image: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 18)

Why MongoDB suits us better?

4. Querying and Visualizing Data:[21]

MongoDB is not limited to simple Key-Value operations. Developers can build rich applications using complex queries, aggregations and secondary indexes that unlock the value in structured, semi-structured and unstructured data, which would be invaluable to our business. A key element of this flexibility is MongoDB's support for many types of queries such as:

- **Key-value queries** return results based on any field in the document, often the primary key
- **Range queries** return results based on values defined as inequalities (e.g, greater than, less than or equal)
- **Geospatial queries** return results based on proximity criteria, intersection and inclusion as specified by a point, line, circle or polygon
- **Text Search queries** return results in relevance order based on text arguments using Boolean operators
- **Aggregation Framework queries** return aggregations of values returned by the query (e.g., count, min, max, average, similar to a SQL GROUP BY statement). Through the $lookup stage, documents from separate collections can be combined through a left outer JOIN operation
- **MapReduce queries** execute complex data processing that is expressed in JavaScript and executed across data in the database

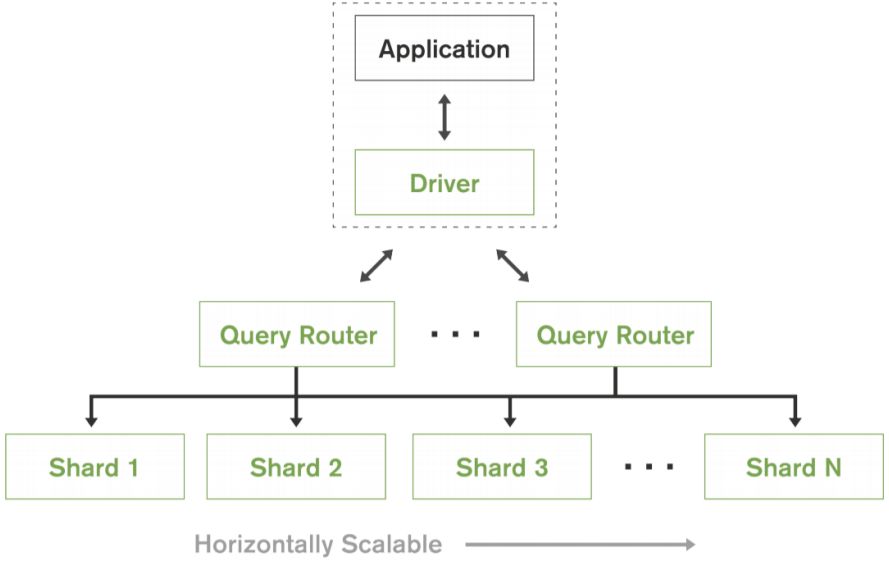
21. Querying and Visualizing Data: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 19)

Why MongoDB suits us better?

5. Auto-Sharding [22] :

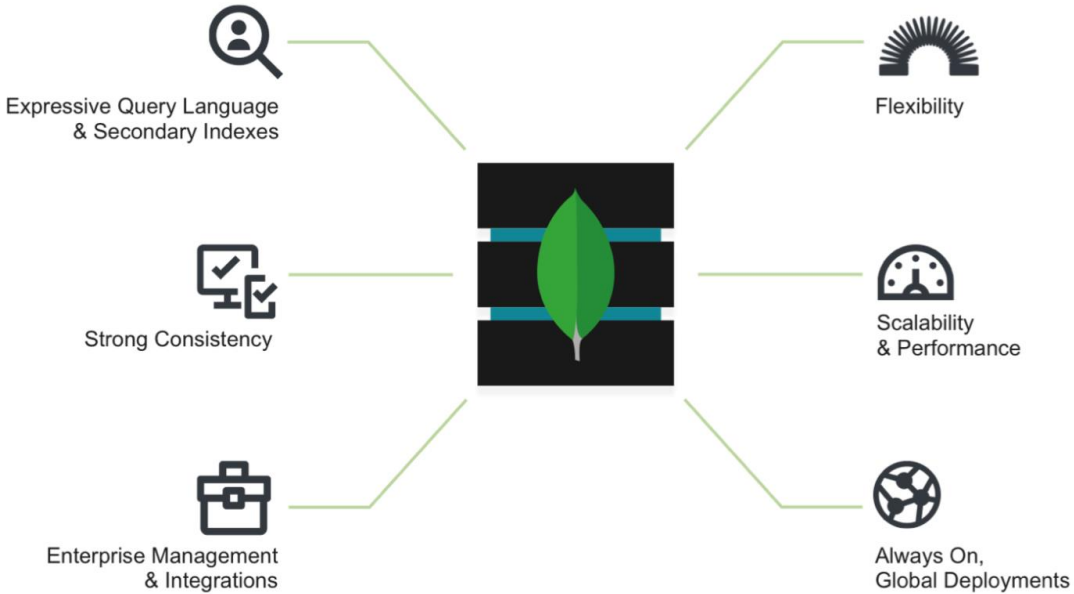
MongoDB provides horizontal scale-out for databases on low cost, commodity hardware or cloud infrastructure using a technique called sharding, which is transparent to applications. Sharding distributes data across multiple physical partitions called shards. Sharding allows MongoDB deployments to address the hardware limitations of a single server, such as bottlenecks in RAM or disk I/O, without adding complexity to the application. **MongoDB automatically balances the data in the sharded cluster as the data grows or the size of the cluster increases or decreases.**

This feature is immensely helpful as StyleUp is still growing and we need to address scaling up as well as scaling down seamlessly



22. Auto-Sharding: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 20)

MongoDB Architecture



MongoDB Nexus Architecture, blending **the best** of relational and NoSQL technologies

23. MongoDB Architecture: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 21)

Expressive Query Language and Secondary Indexes[24]

Users can access and manipulate their data to support both:

Operational applications: StyleUp creates custom orders based on our client's style profile by leveraging our inventory-matching algorithm and inputs from stylists which requires expressive query language & secondary indexes capability of MongoDB to provide efficient access to data, supported natively by the database rather than maintained in application code.

Analytical applications: StyleUp can effectively leverage analytics by tapping into customer feedback obtained from our Chatbot module and review our brand collaborations, performance of our algorithm, monitor changing customer preferences. This requires complex querying enabled by MongoDB as our analytics combines various nuances

24. Expressive Query Language and Secondary Indexes:
https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 22)

Strong Consistency[25]

We can understand our customers better by immediately capturing changes in their style preferences. MongoDB enables StyleUp to immediately read and use data that has been written to the database. It reduces the complexity involved in building applications around an eventually consistent model.

25. Strong Consistency: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 23)

Enterprise Management and Integrations[26]

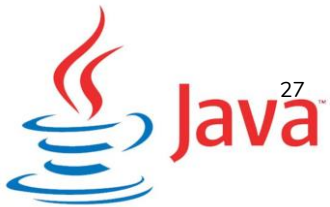
Databases are just one piece of application infrastructure. At StyleUp we require a database solution that fits seamlessly into the enterprise IT stack. MongoDB imposes requirements not addressed by relational databases like MySQL by offering

- **Flexible Data Model:** Document, graph, key-value, or wide-column, all of them offer make it easy to store and combine data of any structure and allow dynamic modification of the schema without downtime or performance impact
- **Always-On Deployments:** Designed to run across many nodes, including replication to automatically synchronize data across servers, racks, and data centers.
- **Scalability and Performance:** Sharding or partitioning allows the database to scale out on commodity hardware deployed on-premises or in the cloud, enabling almost unlimited growth

26. Enterprise Management and Integrations: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 24)a

Upstream

- **Apache:** For back-end GUI Support
- **Java:** Allows creation of Javascript queries
- **MapReduce:** Provides data aggregation capabilities
- **WiredTiger:** The default WiredTiger storage engine for granular concurrency control and native compression will provide the best all round performance and storage efficiency for the broadest range of applications



27

28

29

30

27. Java Image: https://www.theverge.com/2016/1/28/10858250/oracle-java-plugin-deprecation-jdk-9 (Slide 25)
28. Mapreduce Image: https://medium.com/@tejasghalsasi/hadoop-101-getting-started-5d27e7210bb7 (Slide 25)
29. Apache Image：
https://www.google.com/search?biw=1280&bih=617&tbm=isch&sa=1&ei=9tOGXNHgCc2gswWXg6foAQ&q=apache+logo&oq=apache+logo&gs_l=img.3..35i39j0l9.18573.19058..19497...0.0..0.123.443.0j4.....1....1..gws-wiz-img.VK8mlEr4vqY#imgrc=LR0wjo43C1aZ8M:
30. Wiredtiger Image: https://www.slideshare.net/wiredtiger/wired-tiger-overview3 (Slide 25)

Downstream

33

31

32

- **Compass** - Provides GUI support for data visualization, also API support for 3rd party management software
- **Atlas** - Provides cloud support and management for a distributed virtualized shard deployment
- **Stitch** - Back end support and integration for external applications along with API access, also provides application management services

31. Atlas: https://code.tutsplus.com/tutorials/create-a-database-cluster-in-the-cloud-with-mongodb-atlas--cms-31840 (Slide 26)
32. Compass: https://www.kenwalger.com/blog/nosql/mongodb/mongodb-compass-an-overview/ (Slide 26)
33. Stitch: https://medium.com/@nparsons08/mongodb-stitch-your-application-backend-delivered-as-a-service-7cf21d979ed (Slide 26)

High-Level System Requirements

High-Level Requirements

Functional Requirements:

- The system must integrate with the existing StyleUp MySQL Database (DBA)
- The system should have a maintained permissions file using MongoDB “Users and Roles” that is kept at parity with our MySQL database (DBA)
- The system should store all data in a document format as a binary representation called BSON (Binary JSON) (DBA) [34]
- The system must refresh a client’s 360 degree style profile on demand (Stylist DBA)
- The system must support flexible schema design to support our dynamic data sources and data gathering techniques (DBA)
- The system should support document validation as dynamic schemas bring great agility, but it is also important that controls can be implemented to maintain data quality and integrity (DBA) [35]

34. Document format as a binary representation called BSON: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 28)
35. Document validation as Dynamic schemas: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 28)

High-Level Requirements

Functional Requirements:

- The system should support the large volume of information collected from each client and vendor (BM)
- The system should support varied data formats from chatbots conversation in DB such as pictures, text, GIFs, Video etc (DA)
- The system should support key-value queries, range queries, geospatial queries, text search queries, aggregation framework queries and MapReduce queries (DA) [36]
- The system should be able to query and analyze each user interaction across full customer journey by using data to improve engagement and retention (DA)
- The system should be able to schedule analytical reports to communicate business performance in terms of different metrics (DA)

36. The system should support key-value queries..: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 29)

High-Level Requirements

Non-Functional Requirements:

Performance:[37]

- The system must support for many types of secondary indexes that can be declared on any field in the document,including fields within arrays to facilitate faster querying (DA)
- The system must ensure no data lag longer than 15 minutes (CRM/DA/SDBA/BM)
- The system must minimize data latency and optimum utilization of hardware resources (DBA)
- The system should monitor system metrics and send custom alerts before the system degrades (DBA)

Scalability:[37]

- As our data is expected to grow exponentially the system must support it efficiently by sharding (DBA)
- The system must scale in line with scalability of hardware resources
- The system should optimize in case of loss of business and should scale down effectively

37. Performance and Scalability: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 30)

High-Level Requirements

Non-Functional Requirements:

Recoverability:[38]

- The system should support Point-in-time, Scheduled Backups to restore complete running clusters to any point in time with just few clicks, because disasters aren't predictable (CRM/DBA/DA/SDBA)

Concurrency:

- The system should be able to support concurrent user queries (DA)

Security:[38]

- The system should simplifying access control to the database and should specify user-defined roles to enable administrators to configure granular permissions (DBA)
- The system should support data encryption on the network and on disk (DBA)

38.Recoverability and Security: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 31)

High-Level Requirements

Non-Functional Requirements:

Usability:

- The system interface should be easy to learn, navigate and, should not intimidate or overwhelm the user with its complexity. (DBA AND SDBA)
- System has to be designed in a way which will require minimal efforts from the users helping them to focus on achieving goals quickly with few or no errors. (EVERYONE)

Reliability:

- The system should be able perform efficiently without any discrepancies in the result and should have no or minimal amount of downtime in case of catastrophic events.

Interoperability:

- The system should have the capability to interact with third party applications and softwares.

Personas

Pam, the Homemaker

Bio

Pam, 38 was a sales manager at Target, Los Angeles. She is a dedicated mom now who spends her entire day looking after her 3 year old daughter and two year old son. Hence she has very little time to go out and shop for herself or her kids. Her husband is a businessman who cannot spare much time for his family. Michelle is an avid shopper and is very much aware of the changing fashion trends. She loves to shop online.

Qualities:

- Mom of 2 and hence very busy
- Loves Shopping
- Fashion forward

Motivations:

- Quality
- Latest fashion trends
- Value for money

Important Needs:

- She needs a one stop shop
- She needs to manage her changing size issues post pregnancy
- She needs easy returns policies

39. Homemaker Image: https://bit.ly/2UZCDaO (Slide 34)

Ryan, the College Student

Bio

Ryan, 22 is a graduate student at CMU. He is a hardworking and determined student He loves to ride bikes and go on treks. He has a very active social life and loves partying. Also, he is the president of the international students foundation at CMU. He works on campus to cover his expenses.

Qualities:

- Ambitious
- Adventurous and Athletic
- Outgoing and Social

Motivations:

- Comfort
- Ease of buying
- Value for money

Important Needs:

- He needs to shop very frequently but does not want to travel for the same
- He needs a variety of clothes for different events
- He needs to buy clothes at an affordable price

40. College Student Image: https://bit.ly/2TY0Az6 (Slide 35)

Jim, Database Administrator

Bio

Jim, 32 heads our Database Management Team. He is a Computer Engineer from Georgia Tech and is adept at implementing new technologies and his previous experience with McKinsey Project Management team has equipped him with transformation experience. He has strong leadership skills and enjoys taking up challenging projects. He has been instrumental in StyleUp's growth so far.

Qualities:

- Passionate about new technologies
- Great leadership skills

Motivations:

- Contributing to StyleUp's growth
- Opportunity to demonstrate strong database management skills

Important Needs:

- Cost-effective database that supports semi-structured and unstructured data
- Auto-sharding to optimize and automate scaling operations given his small team size
- Document based storage to support StyleUp's Chatbot development and integration

41. Database Administrator Image: https://bit.ly/2SW3I1h (Slide 36)

Kelly, Style Manager DBA

Bio
Kelly, 26, a part of the DBA Team but mainly coordinates with the Stylists to implement their requirements in the Database and facilitates their needs. She is very passionate about the fashion industry and is very skilled at managing latest technologies and database applications.

Qualities:

- Keen eye for detail
- Quick learner and great communicator

Motivations:

- Leverage her unique position as a mediator between DBA team and Stylists to improve StyleUp's product proposition

Important Needs:

- Wants to model the database to facilitate smooth flow of feedback from customers to Stylists
- Minimize latency in the system to ensure timely reporting

42. Style Manager Image: https://bit.ly/2tvXHKI(Slide 37)

Meredith, Data Analyst

Bio

Meredith, 25, is a Statistics graduate from NYU and has been with StyleUp since a year now. Meredith specializes in big data analysis and data visualization. She loves working in startups because of the nature of work which requires her to wear many hats. She believes she will learn more this way. Meredith's determination and curiosity to learn new things has helped StyleUp immensely. She is definitely a valuable asset to our company because of her skills.

Qualities:

- Passionate about data
- Skilled in big data analytics and visualizations
- Driven by her determination and curiosity to learn new advancements in technology

Motivations:

- Make full use of the available data to generate meaningful insights for the executives so that they can make an informed decision.

Important Needs:

- Meredith wants to access the huge amount of historical data for studying consumer behavior and patterns to help the management make informed decisions, with minimum latency

43. Data Analyst Image: https://bit.ly/2NhDzV6 (Slide 38)

Dwight, Brand Manager

Bio
Dwight, 35 has an MBA from The Kellogg School of Management at Northwestern University. Prior to StyleUp, he worked as a Retail Liaison Manager at Macy's. He is responsible for StyleUp's collaborations with different brands and we have benefited tremendously from his market knowledge, relationship with retailers and passion for delivering a fashion forward products.

Qualities:

- Keen eye for detail
- Great at foreseeing market trends

Motivations:

- Ensure StyleUp collaborates with the most popular retail brands
- Discover nascent brands which have great potential

Important Needs:

- Understand our client's brand preferences
- Stay ahead of competition by collaborating with nascent brands which can cater to our client's style profiles
- Get frequent client feedback about brand performance

44. Customer Relationship Manager Image: https://bit.ly/2V5RMXU (Slide 39)

Toby, Customer Relationship Manager

Bio[46]

Toby, 42, is a customer relationship manager at StyleUp. His main responsibilities include increasing the customer base by grabbing all marketing opportunities. He is also responsible for customer retention and loyalty to increase lifetime of customers. He also develops testing strategies for all aspects of the CRM to ensure most effective approach for the company and its customers.

Qualities:

- Customer focused
- Loves to talk to customers and helps them in solving their issues

Motivations:

- Increase customer base, customer loyalty and retention

Important Needs:

- He needs to access historical information efficiently to view the issues that his customers are facing to resolve them quickly
- Wants a one-stop view of all information related to clients such as a 360 degree view
- He wants to provide 24x7 customer support by implementing a chatbot

45. Customer Relationship Manager Bio: https://www.charterselection.com/marketing-job-descriptions/crm-manager-job-description (Slide 40)

47

Michael, Chatbot Developer

Bio

Michael, 28, is a software developer at StyleUp. He has done his Masters in Computer Science from UC Berkeley. He has worked with companies like Doordash to develop their chatbot module. He is proficient in programming languages such a Java, python and SQL.

Qualities:

- Detail oriented
- Has good foresight about scope of the application

Motivations:

- Develop a highly intuitive chatbot module

Important Needs:

- His needs include a database that is robust enough to support different file formats and data sources
- Easy interface and available APIs for reducing connectivity issues
- Low data latency to maintain coherent chatbot conversations

47. Chatbot Developer Image: https://bit.ly/2UudDbN

User Stories
And
Acceptance Tests

User 1: Database Administrator

EPIC

“As a database administrator, I want a database that supports semi-structured and unstructured data in a cost-efficient way, so that data from diverse sources can be organized as documents and it is easier to manage scaling operations.”

User Stories by Database Administrator:

User Story	Acceptance Test
"As a database administrator, I want a database that supports easy integration with our existing database, so that we can complement existing data with new data collected."	1. Ability to perform ETL operations which can support MySQL to JSON data conversion 2. New database should have the ability to create views which translate tabular data into object type data format 3. The above integration process should be condensed and completed in 14 days
"As a database administrator, I want to import existing user and roles along with their access privileges from MySQL to the new database, so that I can maintain consistent level of security and integrity."	1. The new database should be able to ingest the exported users along with their access privileges from MySQL. 2. The above data privilege ingestion process should be completed in 2 days.

User Stories by Database Administrator:

User Story	Acceptance Test
“As a database administrator, I want a database that can store diverse data types in a document format, so that I can facilitate better querying performance since the database needs to read and write to a single location.”	The new database should reduce the latency to no more than 3 seconds.
“As a database administrator, I want a database which will support real-time schema changes without disrupting the existing dependencies and querying capabilities of the system, so that it is easier to scale the system by having dynamic data sources with data in diverse format.”	The new database should support implementation and testing of schema changes in no more than 2 days.

User Stories by Database Administrator:

User Story	Acceptance Test
“As a database administrator, I want a database that provides the capability to perform schema validation during updates and insertion, so that I can implement controls to maintain consistent data quality and integrity.”	The new database should apply validation checks on per collection basis with customizable validation levels and validation actions to minimize data integrity issues by at least 50%.[48]
“As a database administrator, I want the database to monitor system metrics and send custom alerts before the system degrades, so that we can assess the state of the database and resolve bottlenecks if identified.”	The database should support data monitoring tools which will show the users real-time metrics such as:[49] • Reads per second • Writes per second • Read/Writes in progress • Read/Writes in queue • Replication Lag • Replication headroom • No. of transactions per second • Resource Utilization • Resource Saturation • Errors with their status

48.https://docs.mongodb.com/manual/core/schema-validation/
49.https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger/

User Stories by Database Administrator:

User Story	Acceptance Test
“As a database administrator, I want a database which can be scaled along with hardware resources in a way that it distributes data across multiple physical partitions, so that I can address the hardware limitations of single server such as bottlenecks, disc I/Os without adding complexity to the application.”[50]	The current database throughput and availability should remain consistent through the scaling process.[51]
“As a database administrator, I want a database that can has the capability to integrate with third party applications and softwares, so that it can support chatbot and analytics module. ”	The database should minimally support system drivers such as python, NodeJS and Java. [52]

50. https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf
51. https://docs.mongodb.com/manual/sharding/
52. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/

User 2: Chatbot Developer

EPIC

“As a chatbot developer, I want a database that is robust enough to support different data types in a document format and can be easily integrated with my code base, so that I can develop a rich chatbot application which has access to a vast pool of customer data.”

User Stories by Chatbot Developer:

User Story	Acceptance Test
“As a chatbot developer, I want a database which will support unstructured data in document format, so that it can retrieve the data with minimal latency by reading and writing from a single place.”	The new database should reduce the latency to no more than 3 seconds.
“As a chatbot developer, I want a database which will support python, java and NodeJS drivers so that I can integrate it with my chatbot application.”	The database should minimally support system drivers such as python, NodeJS and Java. [53]

53. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/

User Stories by Chatbot Developer:

User Story	Acceptance Test
“As a chatbot developer, I want a database which will support advanced data types not supported by regular text based JSON in MySQL drivers so that I can effectively store records of communications with the clients efficiently. ”[54]	The database should support Binary supported JSON (BSON) format.
“As a chatbot developer, I want a database which offers schema validation based on JSON schema IETF standard so as a developer I can enforce a prescribed data structure for each mongodb collection to streamline data collected from our chatbot.”[55]	The database should support schema validation based on JSON IETF standard.

54. https://www.mongodb.com/compare/mongodb-mysql
55. https://www.mongodb.com/compare/mongodb-mysql

User 3: Data Analyst

EPIC

“As a data analyst, I want a database which will support historical data storage, sourced from diverse platforms, so that I can analyze the patterns and trends in customer behavior to help the management take informed decisions.”

User Stories by Data Analyst:

User Story	Acceptance Test
“As a data analyst, I want a database which will support complex query aggregations, so that I can unlock the value in structured, semi-structured and unstructured data.”	The database solution should minimally support different types of queries such as key-value, range, geo-spatial, text-search, aggregation framework and mapreduce queries. [56]
“As a data analyst, I want a database which will support indexing procedures and has a storage engine which compresses indexes to mitigate their associated overhead so that I can retrieve and analyze the data with minimal latency.”	The database should support unique, compound, array, time-to-live, geo-spatial, partial, sparse and text-search indexing and has an optimized storage engine which compresses indexes. [57]

56. .https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf
57. https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf

User Stories by Data Analyst:

User Story	Acceptance Test
“As a data analyst, I want a database which will support downstream tools for visualizing and analyzing real-time performance metrics, so that I can effectively present my insights.”	The database should have a downstream visualization tool support to informative dashboard creation and to reduce the data refresh lag time to no more than 15 mins.
“As a data analyst, I want a database which supports concurrent user queries, so that database write queues are reduced while maintaining the data integrity.”[58]	The database should use multi-granularity locking system that allows operations to lock at global database or collection level allowing individual storage engines to implement their own concurrency control below the collection level.[58]

58. https://docs.mongodb.com/manual/faq/concurrency/

User 4: Style Manager DBA

EPIC

“As a Style Manager DBA, I want a database that enables me to provide Stylists with a one-stop platform to get a 360 degree view of clients, so that they can understand clients better and customize their packages effectively.”

User Stories by Style Manager DBA:

User Story	Acceptance Test
“As a style manager DBA, I want a database which can associate style tags to each inventory item in an unstructured format, so that the style matching algorithm can efficiently match inventory to client’s preferences.’’	The database should support custom order creation with execution time not exceeding 5 seconds.
“As a style manager DBA, I want a database which will support 360 degree profile creation by pulling the data from varied data sources such as historical customer conversations, client style profile and social media accounts, so that I can provide an one-stop view to the stylist.”	The 360 degree view should help StyleUp achieve return rates of 20% which is 10% below the online retail ecommerce market and it should increase package frequency and package volume by at least 10% [59]

59. https://www.cnbc.com/2016/12/16/a-260-billion-ticking-time-bomb-the-costly-business-of-retail-returns.html

User 5: Brand Manager

EPIC

“As a brand manager, I want to view current & historical customer feedback on all the brands, so that I can analyze the performance of each brand.”

User Story by Brand Manager:

User Story	Acceptance Test
"As a brand manager, I want the database to schedule a periodic report by summarizing current and historical customer feedback associated with each brand stored as a separate document and integrated with the chatbot module, so that I can efficiently monitor each brand's performance	The analytics module should help StyleUp to generate and analyze meaningful reports from historical data which would help the team to make informed decisions. This should help us achieve merchandise turnover rate of 3 vs industry average of 3.91 [60]
"As a brand manager, I want the database to store data and customer sentiment from each prospective partner brand's social media accounts as a separate document and monitor a brand's merchandise turnover rate, so that I can try to include them as our brand partners if they're not already."	The analytics module should help StyleUp to generate meaningful reports on customer's preferred brands, so that brand managers can take further steps to include those brands at StyleUp based on their demand and popularity. We should constantly phase out/review/substitute brands with merchandise turnover rate of more than 4.

60. https://smallbusiness.chron.com/average-merchandise-turnover-clothing-stores-18292.html

User 6: Customer Relationship Manager

EPIC

“As a customer relationship manager, I want to see what common issues my customers have been facing, so that I can get my team to resolve them for improving customer satisfaction index.”

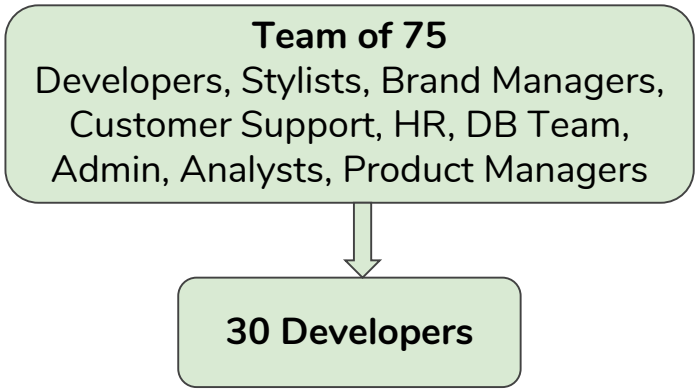
User Story by Customer Relationship Manager:

User Story	Acceptance Test
“As a customer relationship manager, I want a database that facilitates the data analyst to develop a dashboard showing common customer complaints and grievance resolution metrics by querying each customer’s chatbot conversation document, so that I can get my team to resolve them for improving customer satisfaction index.”	The analytics module which is to be included in StyleUp should help the CRM team to view the common and independant issues across all the customers so that they can be resolved quickly. This should help us improve our CSAT index by at least 25%
“As a customer relationship manager, I want a database that can be integrated with the chatbot for my customers to have a medium to interact with us 24x7, so that we are able to help them faster with reduced response waiting time.”	The chatbot module in StyleUp should interact with the customers 24x7 and should generate tickets for the registered complaints so that the CRM team can resolve them in time. This should reduce the average grievance resolution time by at least 30%. The database should minimally support system drivers such as python, NodeJS and Java.[61]

61. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/

Software Development Lifecycle Model

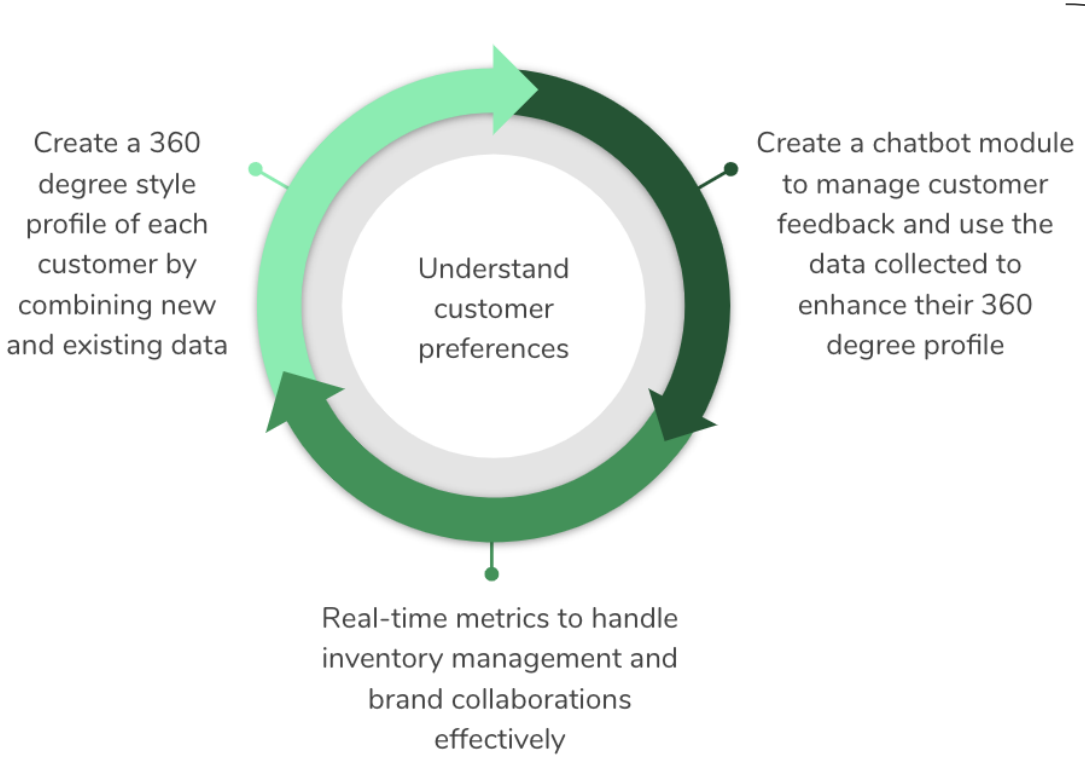
StyleUp Resources



We have recently received Series A funding of $15 Million, out of which we plan to allocate 40% towards hiring 15 new developers, expand our Brand Liaisoning Team and bring in Instagram influencers to guide our team of Stylists

SDLC Model for StyleUp

StyleUp's Proposed Changes involve implementing the following in the next 6 months:



Takeaway:

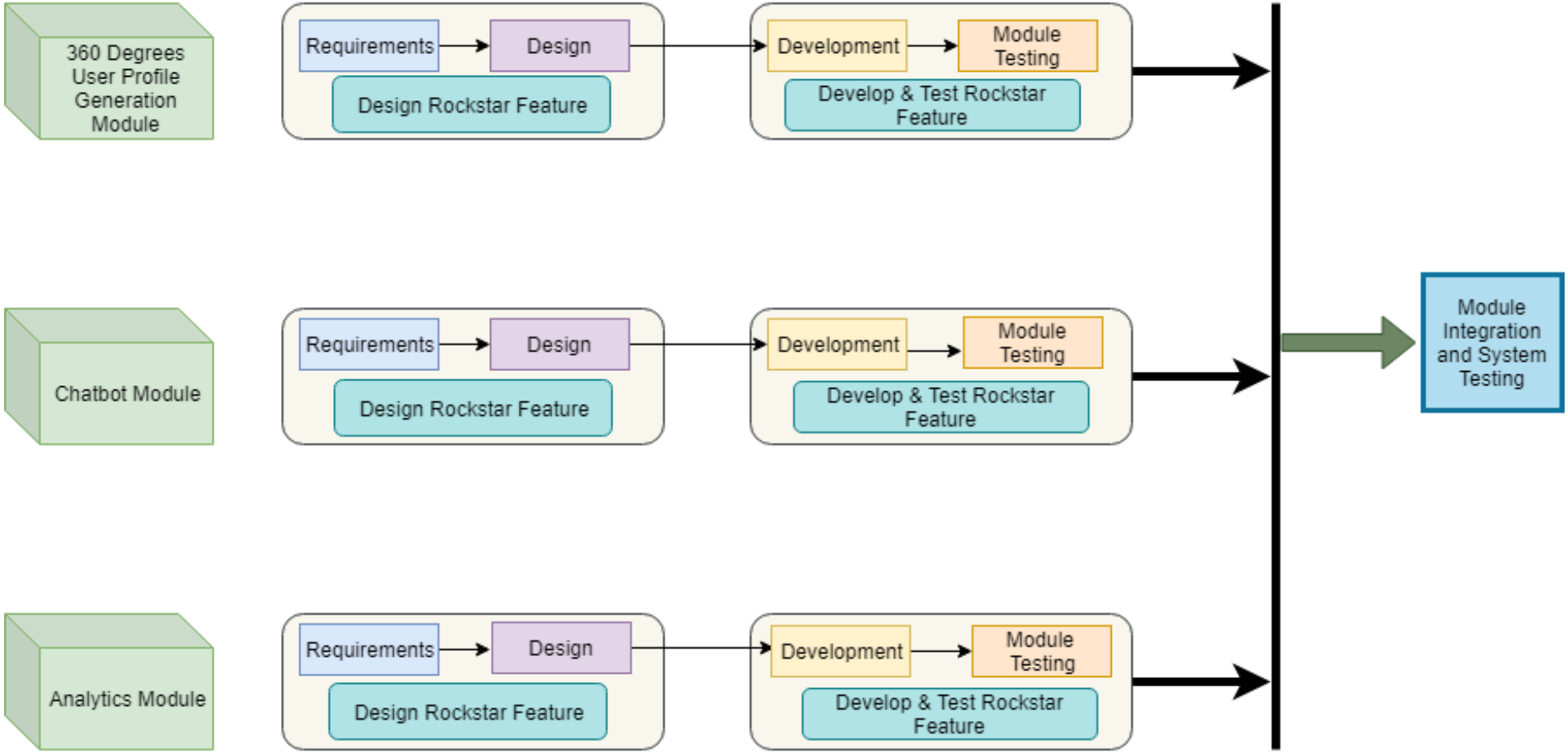
- Action Plan can be **modularized**
- We have one key feature underlying all 3 modules: **Capturing unstructured data and storing it as a document**

SDLC Model for StyleUp

We believe that **Rapid Application Development (RAD) along with Rockstar** would be the most suitable SDLC Model for the following reasons:

- We have 1 dominant Rockstar feature required: **Capturing unstructured data and storing it as a document**
- We have 3 distinct modules which make RAD an ideal model:
  - 360 degree client view to display all information (structured and unstructured) related to a client as a one-stop view
  - Chatbot module to manage customer feedback
  - Real-time Analytics to monitor different metrics
- Since we plan to **implement these new changes over a short span of 6 months**, RAD would be appropriate since we can assign the three modules to different teams which can work to build them simultaneously
- Given our **recent funding allocation plan**, we will have approximately $6 Million (40% of 15), to hire 15 developer DBAs and other domain experts which would facilitate allocating manpower resources to 3 RAD teams

RAD and Rockstar SDLC Model:[46]



46. RAD and Rockstar SDLC Model Diagram drawn using: draw.io (Slide 53)

Return on Investment
&
Return on Value

Return on Investment of StyleUp

Particulars	Yr 1	Yr 2	Yr 3
Present Value of Benefits	$ 840,457	$ 2,146,106	$ 5,265,394
Present Value of Expenses	$ 2,418,182	$ 1,619,835	$ 1,510,143
ROI	-188%	25%	71%
Cumulative PV of Benefits	**$**		**8,251,958**
Cumulative PV of Expenses	**$**		**5,548,159**
Cumulative ROI	**33%**		

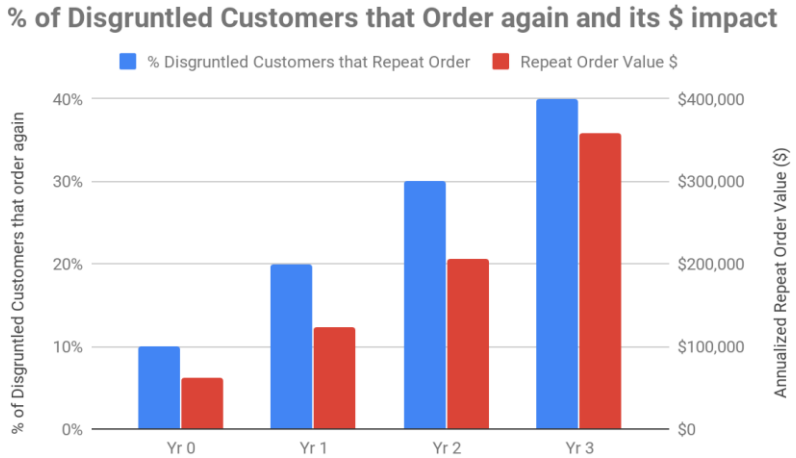
PV of Benefits of StyleUp

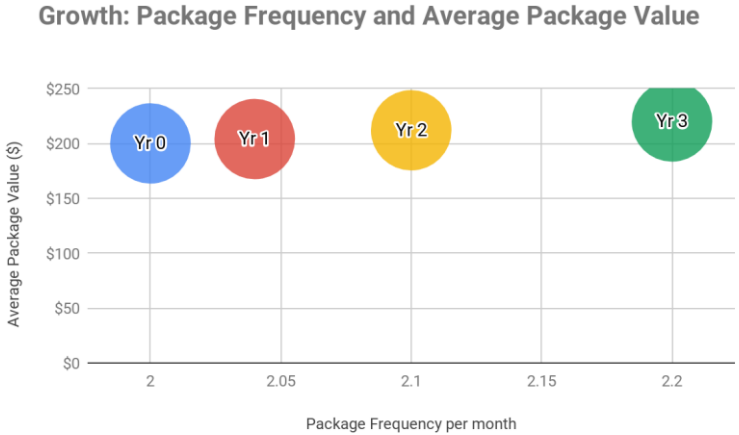
Particulars	Proposed Scenario- Post OSS					
	Base Case	Yr 1	Base Case	Yr 2	Base Case	Yr 3
Shipping Costs						
No of Customers (kept uniform to assess impact)	1,320	1,320	1,500	1500	2,000	2000
Average Shipping Cost per package	$ 14.00	$ 14.00	$ 14.25	$ 14.25	$ 14.50	$ 14.50
Average Package per client per month	2.04	2.04	2.10	2.10	2.20	2.20
Shipping Costs	$ 452,390	$ 452,390	$ 538,650	$ 538,650	$ 765,600	$ 765,600
Return Rates	**30%**	**25%**	**30%**	**22%**	**30%**	**20%**
Return Shipping Costs	$ 120,960	$ 113,098	$ 161,595	$ 118,503	$ 229,680	$ 153,120
Benefit - Reduction in Return Shipping Costs due to Improved Return Rates		**$ 7,862**		**$ 43,092**		**$ 76,560**
Repeat Orders from Disgruntled Customers	**Base Case**	**Yr 1**	**Base Case**	**Yr 2**	**Base Case**	**Yr 3**
No of Customers (kept uniform to assess impact)	1,320	1,320	1,500	1500	2,000	2000
% of negative feedback in a month	19%	19%	18%	18%	17%	17%
Repeat Order Rate from Negative feedback customers	**10%**	**20%**	**10%**	**30%**	**10%**	**40%**
Average $ value per package	$ 204	$ 204	$ 212	$ 212	$ 220	$ 220
Monthly Repeat Order Value from Disgruntled Clients	$ 5,116	$ 10,233	$ 5,724	$ 17,172	$ 7,480	$ 29,920
Annualized Repeat Order Value	$ 61,396	$ 122,792	$ 68,688	$ 206,064	$ 89,760	$ 359,040
Benefit - Repeat Orders from upto 40% customers with negative feedback		**$ 61,396**		**$ 137,376**		**$ 269,280**
Revenue Impact: Package Volume and Frequency growth	**Base Case**	**Yr 1**	**Base Case**	**Yr 2**	**Base Case**	**Yr 3**
Current Customers	1,320	1,320	1,500	1,500	2,000	2,000
Average Package per client per month	**2**	**2.04**	**2.00**	**2.10**	**2.00**	**2.20**
Average $ value per package	**$ 200**	**$ 204**	**$ 200**	**$ 212**	**$ 200**	**$ 220**
Revenue	$ 6,336,000	$ 6,591,974	$ 7,200,000	$ 8,013,600	$ 9,600,000	$ 11,616,000
Benefit - Package Volume and Frequency growth		**$ 255,974**		**$ 813,600**		**$ 2,016,000**
Revenue Impact: Growth in Customer Base	**Year 0**	**Yr 1**	**Year 0**	**Yr 2**	**Year 0**	**Yr 3**
Current Customers	**1,200**	**1,320**	**1200**	**1,500**	**1200**	**2,000**
Average Package per client per month	2.04	2.04	2.10	2.10	2.20	2.20
Average $ value per package	$ 204	$ 204	$ 212	$ 212	$ 220	$ 220
Revenue	$ 5,992,704	$ 6,591,974	$ 6,410,880	$ 8,013,600	$ 6,969,600	$ 11,616,000
Benefit - Growth in Customer Base		**$ 599,270**		**$ 1,602,720**		**$ 4,646,400**
Total Benefit		**$ 924,503**		**$ 2,596,788**		**$ 7,008,240**
PV Factor (10%)		1.1		1.21		1.331
PV of Benefits		**$ 840,457**		**$ 2,146,106**		**$ 5,265,394**
Total Present Value of Benefits	**$**					**8,251,958**

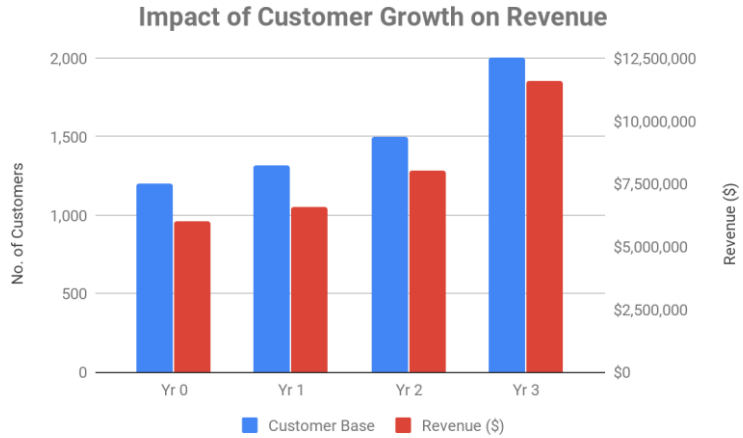
PV of Expenses of StyleUp

Particulars	Yr 1	Yr 2	Yr 3
Cost of New Hires			
Number of Developers to Hire	3	3	3
Per Annum Average Salary	$ 150,000	$ 150,000	$ 150,000
Number of DBAs	6	6	6
Per Annum Average Salary	$ 110,000	$ 110,000	$ 110,000
Number of Data Analysts	6	6	6
Per Annum Average Salary	$ 110,000	$ 110,000	$ 110,000
Total Salary of New Hires	**$ 1,770,000**	**$ 1,770,000**	**$ 1,770,000**
Migration Costs	**$ 500,000**	**$ -**	**$ -**
System Integration Costs	**$ 200,000**	**$ -**	**$ -**
Maintenance Costs	**$ 150,000**	**$ 150,000**	**$ 200,000**
Differential Server Maintenance Costs	**$ 40,000**	**$ 40,000**	**$ 40,000**
Total Expenses	**$ 2,660,000**	**$ 1,960,000**	**$ 2,010,000**
PV Factor (10%)	1.1	1.21	1.331
PV of Expenses	$ 2,418,182	$ 1,619,835	$ 1,510,143
PV of 3 years Expenses	**$**		**5,548,159**

Charts to capture Impact of OSS after Year 0







Return of Value

- **Consistent Delivery:**
  - Integration process of MySQL and MongoDB to be condensed and completed in 14 days
  - Reduce Data Latency to 3 seconds
  - The new database to apply validation checks on per collection basis with customizable validation levels and validation actions to maintain consistency of data [62]
  - MongoDB Auto-Sharding would ensure the current database throughput and availability would remain consistent through the scaling process
  - MongoDB uses a multi-granularity locking system that allows operations to lock at global database or collection level allowing individual storage engines to implement their own concurrency control below the collection level to maintain consistency of data [63]
  - MongoDB consistently supports custom order creation with execution time not exceeding 5 seconds

- **Analysis:**
  - The database should support data monitoring tools which will show the users real-time metrics such as: Throughput, Resource Utilization, Resource Saturation, Errors with their status [64]
  - MongoDB supports different types of queries such as key-value, range, geo-spatial, text-search, aggregation framework and mapreduce queries in addition to secondary indexes to improve query performance and extract better insights for the OSS customers [65]
- **Network:**
  - MongoDB supports system drivers such as python, NodeJS and Java which facilitates downstream applications such as the Chatbot and the Analytics Module [66]

62. https://docs.mongodb.com/manual/core/schema-validation/
63. hhttps://docs.mongodb.com/manual/faq/concurrency/
64. https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger/
65. https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf
66. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/

Testing

Test Plan

Verification Phases

Validation Phases

Detailed Design Analysis

Unit Test Plan

Unit Testing

Functional Specification

Integrated Test Plan

Integration Testing

High Level Design

System Test Plan

System Testing

Requirement Analysis

User Acceptance Test Plan

User Acceptance Testing

Unit Test Plan

Verification Phase

Detailed Design Analysis

Module 1: 360 degree profile view

- Ensure High Level Requirements such as storing data as a BSON document, ability to pull data from various sources are captured in the design phase

Module 2: Chatbot

- Ensure high level requirements such as storage of customer conversations as a document and handling of complex data types are considered in the design phase

Module 3: Analytics

- Ensure high level requirements such as various kinds of secondary indexes, minimum lag times and scheduling custom reports are considered in the design phase

Validation Phase

Module Testing

Module 1: 360 degree profile view

- Given mock customer information, 360 client view should pull and generate a 360 degree profile view
- Test UI
- Test interactions around inventory tags management
- Test that system algorithm is able to interact with the database to create a customized package

Module 2: Chatbot

- Test content classification and capturing diverse data types in the database as a BSON document for a given conversation
- Check past customer order history retrieval flow from the database
- Latency checks for minimum response time

Module 3: Analytics

- Test query performance for each existing metric
- Test lag times in data refresh
- Validate query results accuracy

Integrated Test Plan

Verification Phase

Functional Specification

-Ensuring that each module will work seamlessly with other modules of the system

-Verifying the module integration against the system architecture to ensure they have been integrated as required

-Code will be audited and will be checked if it is producing logically correct output

-Verification of actual test results against expected results after integrating the modules with each other and the hardware

-Proper documentation will be developed, maintained and updated whenever necessary

-It will be ensured that the integration process has not harmed the individual performance of each module

-This phase will also make sure that the data integrity is preserved throughout along with various data manipulation operations

Validation Phase

Integration Testing

End to end flow testing

- Test user style module, chatbot module, 360 degree profile client interactions driven through MongoDB data
- Test customer data like complaints, image, dialog, or feedback from chatbot is factored into 360 degree profile client generation flow
- Test aggregate style data from user style module is factored into 360 degree profile client generation flow
- Test user customized package generation flow with stimulus from both user style module and chatbot module data
- Test analytics flow with user style module, chatbot module, stylist console, 360 degree profile client interactions driven through MongoDB data
- Test input from user style module is factored into analytics report generation flow
- Test input from chatbot module like customer feedback, transcript is factored into report generation flow
- Test feedback and package dispatch requests from style profile are factored into analytics report generation flow
- Test customer profile data from 360 degree profile client view to CRM analytics for report generation

Integrated Test Plan

Verification Phase

Functional Specification

-Ensuring that each module will work seamlessly with other modules of the system
-Verifying the module integration against the system architecture to ensure they have been integrated as required
-Code will be audited and will be checked if it is producing logically correct output
-Verification of actual test results against expected results after integrating the modules with each other and the hardware
-Proper documentation will be developed, maintained and updated whenever necessary
-It will be ensured that the integration process has not harmed the individual performance of each module
-This phase will also make sure that the data integrity is preserved throughout along with various data manipulation operations

Validation Phase

Integration Testing

Test Database interfaces
- Test real time analytics module is able to pull data from MongoDB and generate reports for CRM, Brand Manager and Stylist
- Test 360 degree profile client module is able to pull data from MongoDB and generate 360 degree user profiles
- Test 360 degree profile client module is able to fire request for customized user package vetting
- Test style feedback flows from console to MongoDB
- Test customer complaint ticket data flow into MongoDB
- Test customer feedback data flow into MongoDB
- Test past customer order history retrieval flow from MongoDB to chatbot
- Test user style profile module is able to push aggregated data from social media platforms and user data into MongoDB

System Test Plan

Verification Phase

High Level Design

-In this phase, we will ensure that all the business requirements including functional and non-functional requirements are met.

-All the necessary system documentation will be completed along with detailed verification reports.

-The system will be checked for compliance standards and protocols.

-It will be verified that the system maintains data integrity and consistency throughout the data manipulations.

Validation Phase

System Testing

- Test to ensure that Style Up can support 1000 concurrent users
- Test response time of user queries under low,normal, moderate and heavy load conditions
- Test interactivity with different browsers and platforms
- Test UI interactions are easy to navigate and user friendly
- Test stress to check maximum number of users handled concurrently
-Test data lag response time less than 15 mins
-Test utilization of hardware resources
-System should give positive output for positive tests but should also respond with proper errors or exceptions for negative input

User Acceptance Test Plan

Verification Phase

Requirement Analysis

-Ensuring that all the actions and the operation paths have been documented in the user manual.

-Ensuring that the system not only produces correct output but, it also produces it the correct way.

-Verifying that the user interface is easy to use and navigate and requires minimal user efforts.

-The system meets or outperforms the quality metrics

-Making sure that the system is behaving as per the compliance standards

Validation Phase

User Acceptance Testing

Internal Alpha

- Test mock user style profile creation mimicking user input data
- Test package vetting process with some mock customized user package
- Test user interface renders as expected
- Test chatbot user interface communication gateway and availability 24*7
-Testing the access privilege level for each user
-Validating all business scenarios as defined in design phase

Beta

- UI testing with beta users of all personas, for several iterations to ensure the application is easy to learn and navigate
- Test application on selected browsers to ensure speed and performance

Full Testing Matrix

Test Set	Chrome on WIndow 72.01.3626.121	Chrome on MacOS 72.01.3626.121	Safari on MacOS 12.0.3	Opera 58.0.3135.90	Firefox 65.0.02
Test user style module, chatbot module, 360 degree profile client interactions driven through MongoDB data	Implemented	Implemented	Implemented	Implementation to be completed in next release cycle	Implementation to be completed in next release cycle
Test UI interactions are easy to learn and navigate	Implemented	Implemented	Implemented	Implemented	Implemented
Test performance of the system with respect to load type, users and stress	Implemented	Implemented	Implemented	Implemented	Implemented
Test chatbot user interface is online 24*7	Implemented	Implemented	Implemented	Implemented	Implemented

Bug Tracking and Management

Bug Tracking

- **We are using JIRA for bug tracking and management for our code base ensuring our the bug fix rate to be higher than the bug find rate, causing the number of open bugs to trend downward while keeping the code churn in control**
- **The bug tracking and management will help us with:**[67]
  - ★ Delivering high quality product
  - ★ Improve Return on Investment (ROI) by reducing the cost of development
  - ★ Better communications, teamwork and connectivity across different teams
  - ★ Detecting issues earlier to prevent system disasters
  - ★ Understand and analyze defect trends
  - ★ Improving service quality and customer satisfaction
  - ★ Improve work accountability
  - ★ Work and bug resolvement prioritization
  - ★ Identifying dependencies to reduce the impact of the bug
  - ★ Help plan release checklist better by evaluating the state of the product
  - ★ JIRA will help with automated ticket creation and bug dependency evaluation
  - ★ JIRA will make the Root Cause Analysis and Gap Analysis procedures easier
  - ★ Ensure that the product will behave same under all conditions without errors

67. https://www.testbytes.net/blog/bug-tracking-system/

Bug Tracking

- With help of JIRA and their customizable bug template, we will track the following details for each bug reported:

Bug Description

- ★ Bug ID
- ★ Summary
- ★ Screenshot
- ★ Status: resolved/unresolved
- ★ Severity: critical/major/minor
- ★ Priority Score: P1 to P5
- ★ Dependency Details

Bug Source

- ★ URL: eg. ww.styleup.com/login
- ★ Browser
- ★ Browser version
- ★ Platform
- ★ Operating System
- ★ Steps to reproduce the bug

Administrative Details

- ★ Date
- ★ Assignee
- ★ Assigned to
- ★ Assigned at: timestamp
- ★ Workaround

Release Criteria
&
Release Checklist

Release Criteria [68]

Release criteria are set to meet Quality Assurance measures and will be set before the testing phase to ensure all Acceptance Tests (Slide 44- 59) are met:

- ★ StyleUp chatbot module is able to sustain 1000 active users with latency < 100 ms in load testing
- ★ StyleUp end to end customized package generation flow is able to pass manual stylist approval for at least 10% of trial user data set
- ★ MongoDB is able to ingest unstructured and structured user profile data for a trial group of 1000 users with a delay of < 12 hours
- ★ MongoDB is able to sustain a read QPS > 1 requests/sec to feed the 360 degree client profile pipeline
- ★ 90% of integration tests verifying interface link between MongoDB and real-time analytics will pass
- ★ 90% of integration tests verifying interface link between MongoDB and 360-degree profile client will pass
- ★ 90% of test cases verifying stylist console UI, UI interaction around inventory tags management, feedback flows from console to MongoDB, and package vetting process will pass
- ★ 95% of major severity bugs with priority score P1 are resolved
- ★ 85% of major severity bugs with priority score from P2 to P3 are resolved
- ★ Style Up can handle 1000 concurrent users with a response time of 2 sec
- ★ Style Up is accessible from browsers like Chrome 72.0.3626.121, Safari 12.0.3

68::http://www.professionalqa.com/release-criteria-in-software-testing

Release Criteria

Release criteria are set to meet Quality Assurance measures and will be set before the testing phase to ensure all Acceptance Tests (Slide 44- 59) are met:

- ★ Maximum number and severity of known bugs allowed is:
  - Critical Bugs - 5
  - High Severity Bugs - 7
  - Mid Severity Bugs - 15
  - Low Severity Bugs - 20
- ★ Ratio of new bugs to fixed bugs is 1:4
- ★ All requirements are met and considered in our acceptance tests
- ★ Cut-off time for new bugs is:
  - Critical Bugs - 7 days
  - High Severity Bugs - 10 days
  - Mid Severity Bugs - 14 days
  - Low Severity Bugs - 30 days
- ★ Release approval process is completed thoroughly

Release Checklist

Project	
Release Number	1.2.0.1
Release Audience	Alpha release(Internal testers access) Beta release(Controlled external access

Quality Assurance

Task	Team Responsible
Verify all module test cases for 360-degree client profile, chatbot module, analytics run on the integrated system	Testing
Verify all bugs with major severity and priority score P1to P3 are resolved	Testing
Test Style Up compatibility across browsers like Chrome, Safari on all devices	Testing
Perform consistency test with requirement specification documents, user manual and , software	Testing

Release Checklist[69,70]

Documentation Activities

Task	Team Responsible
Verify user documentation matches requirements in the current release	Development
Update user manual guide for Style Up	Development

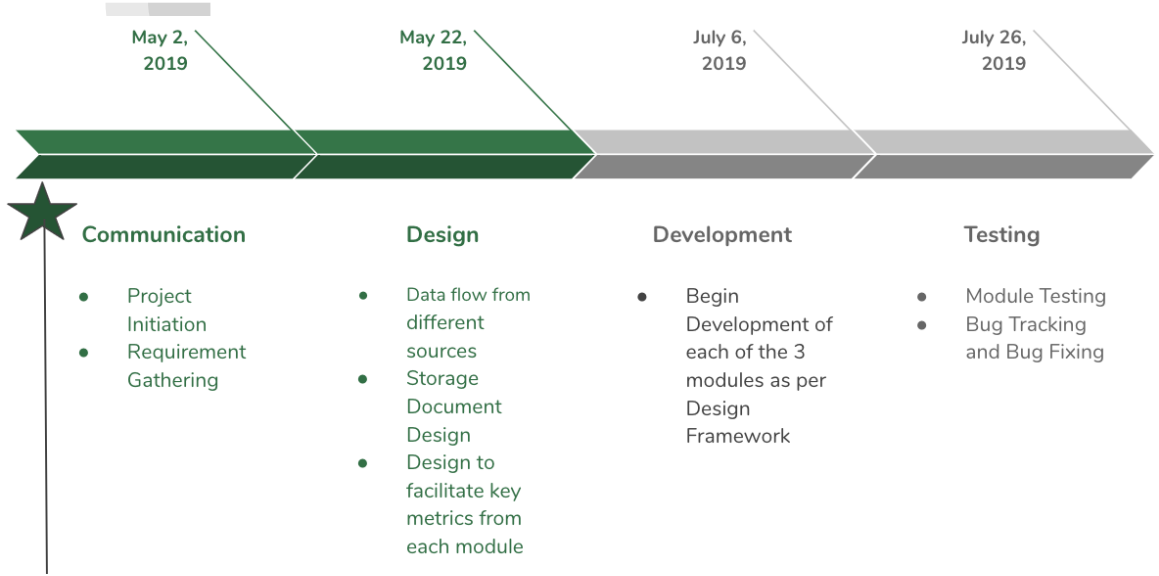
Legal Activities

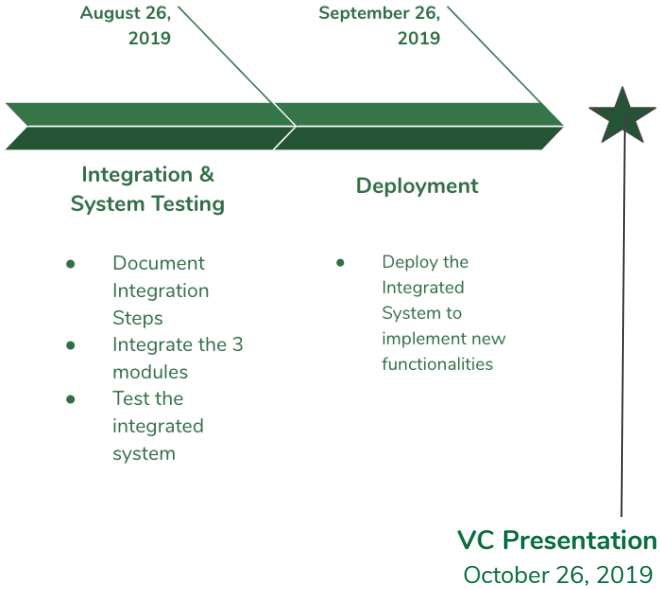
Task	Team Responsible
Verify legal risk associated with the release are identified and resolved	Legal
Verify licenses and intellectual property criteria are met	Legal

69 .http://users.csc.calpoly.edu/~jdalbey/206/Templates/releasechecklist.html
70 .https://people.cs.pitt.edu/~chang/231/seminars/S06template/templates/release-checklist.html

Timeline and Milestones

Timeline





Each of the 3 modules are managed by 3 teams independently in line with the RAD Model

All the 3 Modules are Integrated and Deployed together

Product Roadmap

Phase 1

Prerequisite task

Migrating Data from MySQL to MongoDB

Finalizing client data sources

Phase 2

Development (RAD)

Module 1: 360 degree view

Module 2: Chatbot Integration

Module 3: Analytics

Phase 3

Module Testing

Module 1: 360 degree view

Module 2: Chatbot Integration

Module 3: Analytics

Phase 4

Integration, System and User Acceptance Testing

Release Roadmap

Release Roadmap

Phase 1

Bug Tracking and Management

Defining a New bugs : Fixed bugs ratio

Define an appropriate code churn rate

Prioritize bugs

Triaging incoming bugs

Phase 2

Release Criteria

Max number and severity of bugs defined

Acceptance Tests compliance checked

Mapping requirements

Define cut-off time for bugs

Phase 3

Release Processes

Design Release Checklist

Release Reviews

Release Approvals

Phase 4

Maintenance Framework

Define scope of organizational responsibility to handle user issues

Teams of Dedicated Staff for each of the 3 Modules

Schedule Improvements in design and implementation

Release

Risks & Mitigation

Key Focus Areas

- Identify Risks at the requirements stage
- Ensure the Rockstar feature takes precedence across all 3 modules
- Since the project is implemented using RAD, involving 3 different teams, ensure that all the processes are well-documented
- Ensure the legacy system (MySQL) is operational and isolated from the changes being implemented

Risk Table

Risks	Probability	Severity	Mitigation	Measures for Risk Avoidance
Operational risk in moving to a new technology stack	Low	Medium	Transition such that there is no user impact due to the migration by planning transition gradually and step-wise	Developing a detailed, end to end migration plan addressing all possible issues and providing cross functional training and technical support to team
Risk of data loss with relaxed DB consistency guarantees	Medium	High	Validate assumptions around data consistency **Moved to requirements**	Add data consistency check across database to do proactive bug finding
Resistance to change	Low	Medium	Employees should be given proper training and all employee concerns should be constantly addressed during development process	communication for change in technology is essential and proactively discussing the change requirements and the process would help to build trust.

Risk Table

Risk	Probability	Severity	Mitigation	Measures for Risk Avoidance
Development resource constraints due to migration workload	High	Medium	Plan for staffing resource and commit all product feature related work to align with migration timelines	Develop resource allocation plans and identify critical resources and develop contingency plan for substitution of those resources
Bugs due to less static checking and tooling due to schema-less design	High	Medium	Evolve design practices around structured data to rely more on dynamic validation to ensure data integrity	Develop dynamic validation test cases to ensure data quality
Change in performance characteristics due to DB migration	Medium	High	1) Conducting performance drills and load tests to ensure performance is on par or better than MySQL. 2) Ensuring querying patterns are optimized for MongoDB	Create test cases to validate the migrated database and test against metrics

Risk Table

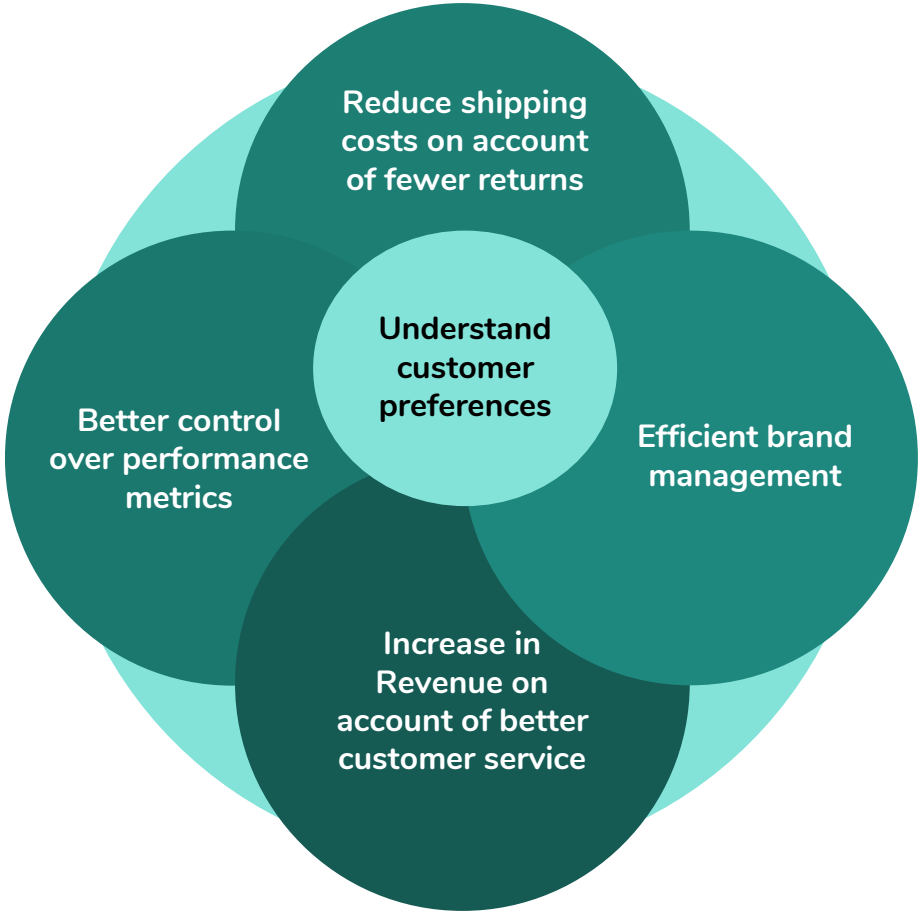
Risk	Probability	Severity	Mitigation	Measures for Risk Avoidance
Securing customer data	Medium	High	1) Review auditing/feedback business process to properly secure customer data 2) Protect against customer data scraping by setting monitoring 3) Encrypting customer sensitive data	Develop detailed customer data protection rules' and identify high risk data and secure by implementing safeguards with regular monitoring and testing

Risk Table

Risk	Probability	Severity	Mitigation	Measures for Risk Avoidance
Showing proper error messages for invalid operations	Medium	Low	All unencountered errors should be reported and documented and added to the system wherever necessary	Document all errors for every combination of operation and add error messages wherever appropriate
Not encrypting open source/public data, that is supposed to be accessed by all	Low	Low	All unencrypted public data should be reported immediately to the concerned department for encryption	Creating a proper encryption system that encrypts all data in the system

Business Value & Success Metrics

Business Value



Success Metrics - Business

25%	• Return Shipping cost reduced by at least 25% over 3 years
15%	• Maintain a return rate of 20% which is 10% below the rate of online retail ecommerce market
40%	• Repeat orders from at least 40% of the customers interacting with our Chatbot to give negative feedback
25%	• Improve our CSAT index by at least 25%
3 m	• Merchandise turnover rate of 3 (months) for each brand vs industry average of 3.91 (months)

Success Metrics - IT

3s	• Database solution adopted should limit latency to no more than 3 seconds
20%	• Average package frequency should increase by 10% over 3 years
20%	• Average package volume should increase by 10% over 3 years
30%	• Reduce the average grievance resolution time by at least 30%

References

References

1. Mission Statement: https://www.stitchfix.com/about (Slide 3)
2. Confused Shopper Image: https://bit.ly/2TVZ4gW (Slide 5)
3. Survey Image: https://bit.ly/2TWxJv1 (Slide 5)
4. Delivery Person Image: https://bit.ly/2BH3gKa (Slide 5)
5. Happy Shopper Image: https://bit.ly/2EgRecs (Slide 5)
6. Brands Image: https://bit.ly/2GvsjUT (Slide 6)
7. Diverse Shoppers Image: https://bit.ly/2DUcx1S (Slide 6)
8. Growth Image: https://bit.ly/2towo4C (Slide 7)
9. Existing System Flowchart drawn using: https://www.draw.io/ (Slide 8)
10. Overwhelmed Person Image: https://bit.ly/2TVjN4B (Slide 9)
11. MySQL Oracle owned, not community driven: https://smartbear.com/blog/test-and-monitor/5-reasons-its-time-to-ditch-mysql/ (Slide 9)
12. MySQL Scalability issue: https://www.gridgain.com/resources/blog/5-limitations-mysql-big-data (Slide 9)
13. Features of MongoDB: https://docs.mongodb.com/manual/introduction/ (Slide 11)
14. Proposed System Flow Chart drawn using: https://www.draw.io/ (Slide 12)
15. Developer productivity is increased with JSON Documents: https://www.mongodb.com/compare/mongodb-mysql (Slide 15)
16. Document storage Image: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 15)

References

17. MySQL's JSON data type does not bring the developer productivity: https://www.mongodb.com/compare/mongodb-mysql (Slide 16)
18. MySQL's JSON data type does not bring the developer productivity: https://www.mongodb.com/compare/mongodb-mysql (Slide 17)
19. MongoDB maintains multiple copies of data: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 18)
20. Replication Image: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 18)
21. Querying and Visualizing Data: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 19)
22. Auto-Sharding: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 20)
23. MongoDB Architecture: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 21)
24. Expressive Query Language and Secondary Indexes: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 22)
25. Strong Consistency: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 23)
26. Enterprise Management and Integrations: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 24)

References

27. Java Image: https://www.theverge.com/2016/1/28/10858250/oracle-java-plugin-deprecation-jdk-9 (Slide 25)
28. Mapreduce Image: https://medium.com/@tejasghalsasi/hadoop-101-getting-started-5d27e7210bb7 (Slide 25)
29. Apache Image：
https://www.google.com/search?biw=1280&bih=617&tbm=isch&sa=1&ei=9tOGXNHgCc2gswWXg6foAQ&q=apache+logo&oq=apache+logo&gs_l=img.3..35i39j0l9.18573.19058..19497...0.0..0.123.443.0j4.....1....1..gws-wiz-img.VK8mlEr4vqY#imgrc=LR0wjo43C1aZ8M:
30. Wiredtiger Image: https://www.slideshare.net/wiredtiger/wired-tiger-overview3 (Slide 25)
31. Atlas: https://code.tutsplus.com/tutorials/create-a-database-cluster-in-the-cloud-with-mongodb-atlas--cms-31840 (Slide 26)
32. Compass: https://www.kenwalger.com/blog/nosql/mongodb/mongodb-compass-an-overview/ (Slide 26)
33. Stitch: https://medium.com/@nparsons08/mongodb-stitch-your-application-backend-delivered-as-a-service-7cf21d979ed (Slide 26)
34. Document format as a binary representation called BSON:
https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 28)
35. Document validation as Dynamic schemas:
https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 28)
36. The system should support key-value queries..:
https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 29)

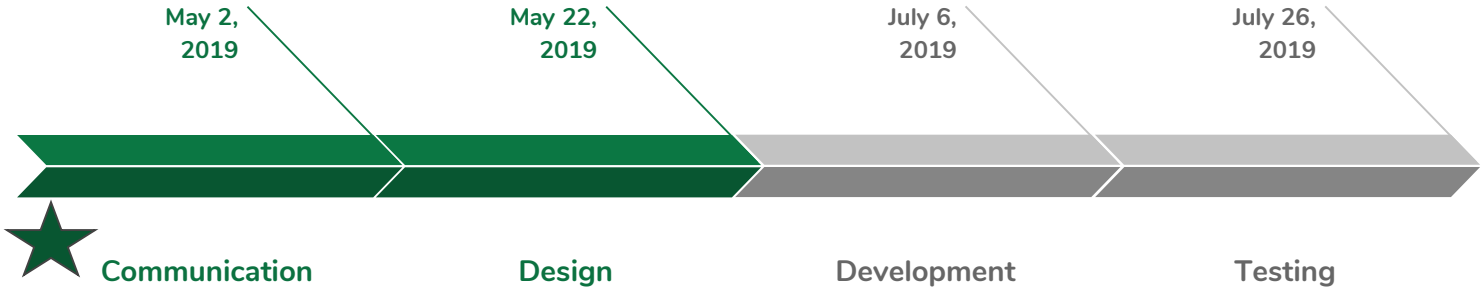
References

37. Performance and Scalability: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 30)
38.Recoverability and Security: https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 31)
39. Homemaker Image: https://bit.ly/2UZCDaO (Slide 34)
40. College Student Image: https://bit.ly/2TY0Az6 (Slide 35)
41. Database Administrator Image: https://bit.ly/2SW3I1h (Slide 36)
42. Style Manager Image: https://bit.ly/2tvXHKI(Slide 37)
43. Data Analyst Image: https://bit.ly/2NhDzV6 (Slide 38)
44. Customer Relationship Manager Image: https://bit.ly/2V5RMXU (Slide 39)
45. Customer Relationship Manager Bio: https://www.charterselection.com/marketing-job-descriptions/crm-manager-job-description (Slide 40)
46. RAD and Rockstar SDLC Model Diagram drawn using: draw.io (Slide 53)
47. Chatbot Developer Image: https://bit.ly/2UudDbN （slide 41）
48.https://docs.mongodb.com/manual/core/schema-validation/ (slide 46)
49.https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger/ (slide 46)
50.https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (slide 47)
51. https://docs.mongodb.com/manual/sharding/ (Slide 47)
52. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/ (Slide 47)
53. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/ (Slide 49)

References

54. https://www.mongodb.com/compare/mongodb-mysql (Slide 50)
55. https://www.mongodb.com/compare/mongodb-mysql (Slide 50)
56. .https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 52)
57. https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 52)
58. https://docs.mongodb.com/manual/faq/concurrency/ (Slide 53)
59.https://www.cnbc.com/2016/12/16/a-260-billion-ticking-time-bomb-the-costly-business-of-retail-returns.html
(Slide 55)
60. https://smallbusiness.chron.com/average-merchandise-turnover-clothing-stores-18292.html (Slide 57)
61.https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/ (Slide 59)
62. https://docs.mongodb.com/manual/core/schema-validation/ (Slide 70)
63. hhttps://docs.mongodb.com/manual/faq/concurrency/ (Slide 70)
64. https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger/ (Slide 70)
65. https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf (Slide 70)
66. https://docs.mongodb.com/ecosystem/drivers/driver-compatibility-reference/ (Slide 70)
67. https://www.testbytes.net/blog/bug-tracking-system/ （Slide 80)
68::http://www.professionalqa.com/release-criteria-in-software-testing (Slide 83)
69. .http://users.csc.calpoly.edu/~jdalbey/206/Templates/releasechecklist.html (Slide 86)
70 .https://people.cs.pitt.edu/~chang/231/seminars/S06template/templates/release-checklist.html (Slide 86)

Backup Slides



Communication

- Project Initiation
- Requirement Gathering

Design

- Data flow from different sources
- Storage Document Design
- Design to facilitate key metrics from each module

Development

- Begin Development of each of the 3 modules as per Design Framework

Testing

- Module Testing
- Bug Tracking and Bug Fixing

Integration & System Testing

- Document Integration Steps
- Integrate the 3 modules
- Test the integrated system

Deployment

- Deploy the Integrated System to implement new functionalities

Thank You!