Software Requirements Specification (SRS) Document

Mobile Ticketing App
Team Number 10
Nishanth Sharma - Divyesh Jain - Prateek Saini

Brief problem statement

We are building (Proof of Concept) a web app for booking tickets. The web app is expected to suggest movies to users based on twitter data. A real-time server does a sentimental analysis on twitter data to get movie reviews. A message broking server gets data from twitter and stores this data in a NoSQL database. The web app sorts all movies based on reviews from the sentimental analysis part and renders a html page suggesting movies in decreasing order of the reviews.

System requirements

- Apache Kafka for message broking server.
- Cassandra Data Base for NoSQL Data Base.
- Flask as a web framework.
- Bootstrap and Jinja template for user interface.
- Apache Spark for real-time analysis of data.
- Text blob API for sentimental analysis.
- SQLite3 for back end.

Users profile

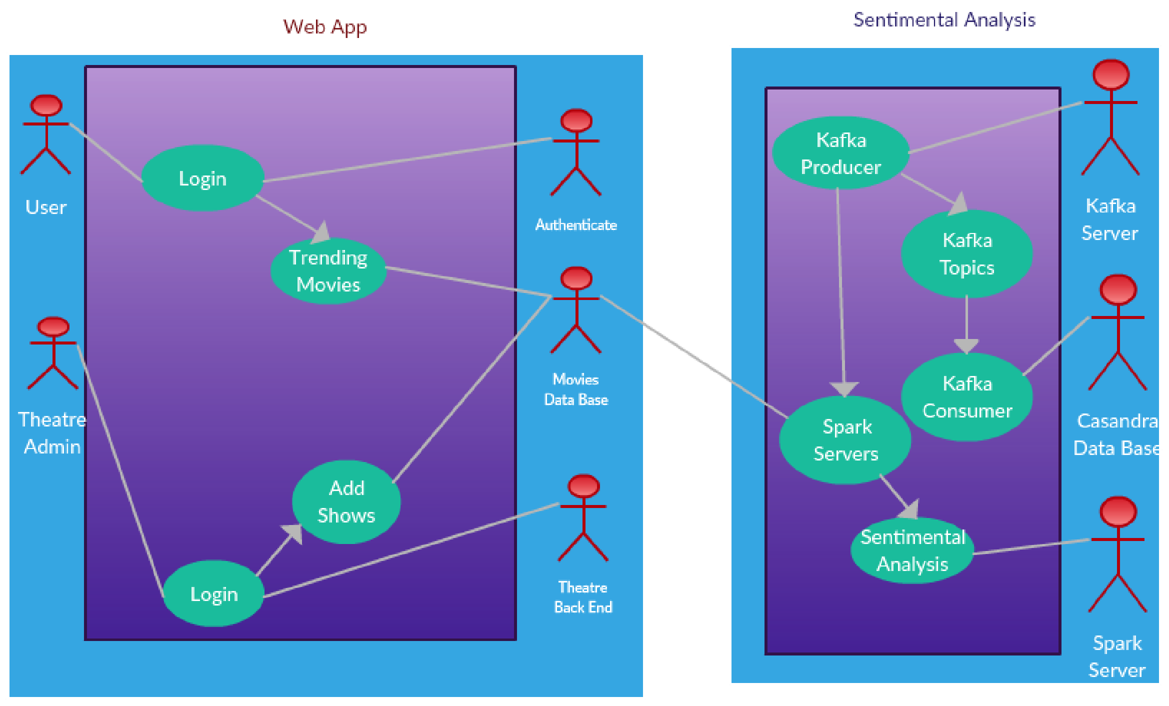
The project is expected to suggest movies to users based on twitter data. And then let user book tickets. Booking tickets is mostly done online. So, the project targets almost everybody. And for a better review we need a lot of data. As twitter is very popular social media platform, we can get very good reviews for movies and suggest movies to users.

Feature requirements (described using use cases)

No.	User Case Name	Description	Release
1.	Suggesting Movies	Trending movies, based on tweets in real-time.	12th week
2.	User Dashboard	Information regarding trending movies.	9th week
3.	Theatre	Interface for a theatre admin to manage information	9th week

	Administration	related to a particular theatre.	
4.	Personal User Account	A personal account for all new users.	7th week
5.	Getting twitter feed	Getting data from twitter and sending to Kafka server.	7th week
6.	Message Broking Server	Getting data stream from Kafka and storing in Cassandra database.	6th week
7.	Spark Server Analysis	Real-time spark server which does analysis on data stream from twitter.	8th week
8.	Multiple Kafka Topics	Run multiple Kafka topics to get twitter data of multiple movies at once	10th week
9.	Multiple Spark Servers	Run multiple Spark Servers to update multiple movie reviews at once.	11th week

Use case diagram



Use case description

Use Case Number:	1
Use Case Name:	Suggesting Movies
Overview:	Movie reviews are updated after small time intervals (sentimental analysis done using the spark server) and are visible as a dynamic list on the user dashboard.
Actors:	Spark Server, SQL Database, Kafka Server
Pre-condition:	Database is working, Spark server is not down and Kafka Server is not down.
Flow:	Main (success) Flow: 1. Spark server gets twitter data stream from Kafka server. 2. Spark server updates review for the movie using tweet. 3. Movie review is updated in the movie (SQL) database.
	Alternate Flows: 1. Spark or Kafka server fails. Post-condition: Non-updated movie reviews are shown on user dashboard.
Post-condition:	Trending movies list keeps getting updated with time as movie reviews keep getting updated.

Use Case Number:	2
Use Case Name:	User Dashboard
Overview:	Webpage used to display information regarding trending movies using the underlying sentimental analysis.
Actors:	User, SQL Database
Pre-condition:	Database is working.
Flow:	Main (success) Flow: 1. User enters their credentials on the login page. 2. User is redirected to the dashboard.
	Alternate Flows: 1. User enters invalid credentials. Post-condition: Error message is flashed.
Post-condition:	User is able to view a list of movies sorted by decreasing order of the reviews given to the movies.

Use Case Number:	3
Use Case Name:	Theatre Administration
Overview:	An interface for an administrator (from theatre staff) can add and edit information about said theatre.
Actors:	Theatre Admin, SQL Database
Pre-condition:	Database is working
Flow:	Main (success) Flow: 1. Theatre admin enters all of the information regarding the theatre on the theatre register page. 2. Theatre admin is redirected to theatre login page. 3. Theatre admin enters credentials and upon authentication, is redirected to the add show page.
	Alternate Flows: 1. Theatre admin enters invalid credentials on the theatre login page. Post-condition: Error message is flashed. 2. Database is not working so theatre admin can't be authenticated at the time. Post-condition: Error message is shown that database is down.
Post-condition:	Theatre admin is successfully registered and can login anytime to modify information regarding their theatre.

Use Case Number:	4
Use Case Name:	Personal User Account
Overview:	A personalized space for the user to view information of their preference, which nobody else has access to.
Actors:	User, SQL Database
Pre-condition:	Database is working
Flow:	Main (success) Flow: 1. User enters his/her details on register page and is successfully registered. 2. User is redirected to the login page. 3. User enters credentials, which are authenticated with the database.
	Alternate Flows: 1. User enters invalid username and/or password on the login page. Post-condition: Flash error message.

	2. Database is down so user credentials cannot be authenticated. Post-condition: Flash message telling user that they can’t be authenticated right now.
Post-condition:	User has successfully created their personal account and can login anytime.

Use Case Number:	5
Use Case Name:	Getting twitter feed
Overview:	Getting data from twitter and sending to message broking server to be sent to message broking server Kafka.
Actors:	Twitter, Kafka Server
Pre-condition:	Twitter Server is not down.
Flow:	Main (success) Flow: 1. Authenticate to twitter developer account. 2. Connect to Twitter Stream Listener. 3. Send data stream to Kafka producer.
	Alternate Flows: 1. Authentication failure. Post Condition: Throw error. 2. Connection failure to Twitter Stream Failure Post Condition: Throw Error 3. Kafka producer failure to send data stream. Post Condition: Throw Error
Post-condition:	Kafka consumer gets data and starts storing twitter data in NoSQL database.

Use Case Number:	6
Use Case Name:	Message Broking Server
Overview:	Kafka consumer receiving data from message broking server and cleaning the messages received. Then storing the messages in Cassandra NoSQL data base.
Actors:	NoSQL database, Kafka Server
Pre-condition:	Kafka Producer is working.
Flow:	Main (success) Flow:

	1. Consume tweets from message broking server (Kafka). 2. Store tweets in NoSQL data base (Cassandra).
	Alternate Flows: 1. Kafka server fails. Post Condition: Wait until a non-null message is received. 2. Failure to connect to Cassandra data base. Post Condition: Throw Error
Post-condition:	Add tweets to Cassandra data base.

Use Case Number:	7
Use Case Name:	Spark Analysis Server
Overview:	Real-time analysis of twitter data and get sentiments of each tweet. Update review in movie data base.
Actors:	Kafka Server, Spark Server
Pre-condition:	Kafka Server is not down.
Flow:	Main (success) Flow: 1. Spark Server receives Kafka data stream from Kafka server. 2. Sentimental analysis of tweets received through Kafka data stream. 3. Update review of movies.
	Alternate Flows: 1. Kafka server fails. Post Condition: Wait until a non-null message is received. 2. Receiving null messages. Post Condition: Ignore Null messages 3. Failure to connect to movies data base. Post Condition: Throw Error.
Post Condition:	Update movie review in SQL data base.

Use Case Number:	8
Use Case Name:	Multiple Kafka Topics

Overview:	Multiple Kafka topics are run at once to update movie reviews in real time. So multiple threads are run Kafka Server to receive multiple movie tweets at once.
Actors:	None
Pre-condition:	Kafka Server is not down.
Flow:	Main (success) Flow: 1. Run multiple Kafka Producer. 2. Run multiple Kafka Consumer.
	Alternate Flows: 1. Kafka Producer fails. Post Condition: Throw Error. 2. Kafka Producer fails. Post Condition: Throw Error.
Post Condition:	Add tweets to Cassandra data base.

Use Case Number:	9
Use Case Name:	Multiple Spark Servers
Overview:	Multiple Spark Servers are run at once. To update movie reviews at once. Each Server is run for certain period of time to keep server use its computing for maximum time.
Actors:	Spark Server
Pre-condition:	Spark Server is working.
Flow:	Main (success) Flow: 1. Get twitter data stream from Kafka server. 2. Run multiple Spark Servers at once.
	Alternate Flows: 1. Kafka server fails. Post Condition: Wait until a non-null message is received. 2. Two process use Spark Server at once. Post Condition: Throw Error
Post Condition:	Update movie reviews in SQL data base.