System Design Specification (SDS) simplyatx.com

Current Version:	1.0 Release
Author(s):	Borislav Sabotinov, Justin Franz, Lanlan Liu
Date Created:	3/5/2020

Revision History

Version Number	Date Updated	Revision Author	Brief Description of Changes
0.1	3/5/2020	Boris	Created document
0.2	4/10/2020	Lanlan	Updated outline for sections 2-5
0.3	4/15/2020	Boris	Added class and architecture diagrams
0.4	4/16/2020	Lanlan	Updated user diagrams
0.5	4/17/2020	Justin	Added use case tables and their details
0.6	4/17/2020	Boris	Added table describing class diagram at the feature level, description of architecture diagram, and updated use case diagrams
0.7	4/17/2020	Lanlan	Edited use case tables
1.0	4/17/2020	Boris	Formatting and finalized final release draft.

Table of Contents

1 Introduction 3
1.1 Purpose of the SDD 3
1.2 System Scope 3
1.3 Definitions and Acronyms 3
1.4 References 4
1.5 Overview 4
2 Use Cases 5
2.1 Actors 5
2.1.1 Administrative user 5
2.1.2 General User with Admin for User Feedback 6
2.2 Use Cases 8
3 Design Overview 14
3.1 Introduction 14
3.2 System Architecture 14
4 System Object Model 15
4.1 Introduction 15
4.2 Subsystems 15
5 Object Descriptions 16
5.1 Class diagram 16
5.2 Description of Attributes and Methods (Client-side code) 19
5.2.1 User Login 20
5.2.2 Perform Search 20
5.2.3 Leave a Review 20
5.2.4 Subscribe to notification 21
5.2.5 Leave Feedback 21
5.2 User Interface Mapping Diagram 22

1 Introduction

1.1 Purpose of the SDD

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built. It also provides a list of requirements against which to test the final project and determine whether the system is successfully implemented according to the design. This SDD outlines the design for the SimplyATX website.

1.2 System Scope

The software product that will be produced is a website called SimplyATX. The URL is https://simplyatx.com/.

SimplyATX will:

- Help users find information about movies and TV listings.
- Provide an interface for users to find a movie or TV listing of their choice.
- Provide an interface for users to rate a movie or TV listing, using a unique and fun emoji scheme.
- Provide an interface for users to buy a movie ticket from an external vendor.
- Provide an interface for users to create and manage a personal user account with limited options.
- Aggregate reviews and showtime information from public API’s that can be leveraged, as well as get relevant showtime information and display it.
- Provide an interface for users to subscribe and receive notifications about predefined categories and special offerings, if possible.
- Provide an interface for users to get contribution milestones.
- Provide an optional interface for users to leave feedback about the website.

The ultimate goal of this website is to give users the ability to find movie information, view and leave reviews, buy tickets for movies, and subscribe to be notified of special offerings.

1.3 Definitions and Acronyms

Table 1: Terms and Definitions

Term	Definition
SRS	System Requirement Specification, this document will outline all the requirements that the software must fulfil.
UI / GUI	User interface / Graphical User Interface
Backend	The part of a computer system or application that is not directly accessed by the user, typically responsible for storing and manipulating data.

Frontend	Relating or denoting the part of a computer system or application with which the user interacts directly
Interface	a device or program enabling a user to communicate with a computer.
API	Application programming interface
IMDB	International Movie Database, a website which provides movie and TV show related information
IE	Internet Explorer, a web browser developed by Microsoft.
WAN	Wide area network. A telecommunications network that extends over a large geographical area for the primary purpose of computer networking.
LAN	Local area network. The SimplyATX application is not accessible via LAN only networks.
SMTP	Simple Mail Transfer Protocol
Session	A user's time in the system measured from login to logout

1.4 References

[1] IEEE Software Engineering Standards Committee, "IEEE Std 1016-2009, IEEE Recommended Practice for Software Design Description"
http://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf

[2] Use Case Diagrams Reference. 2009-2020 uml-diagrams.org
https://www.uml-diagrams.org/use-case-reference.html

1.5 Overview

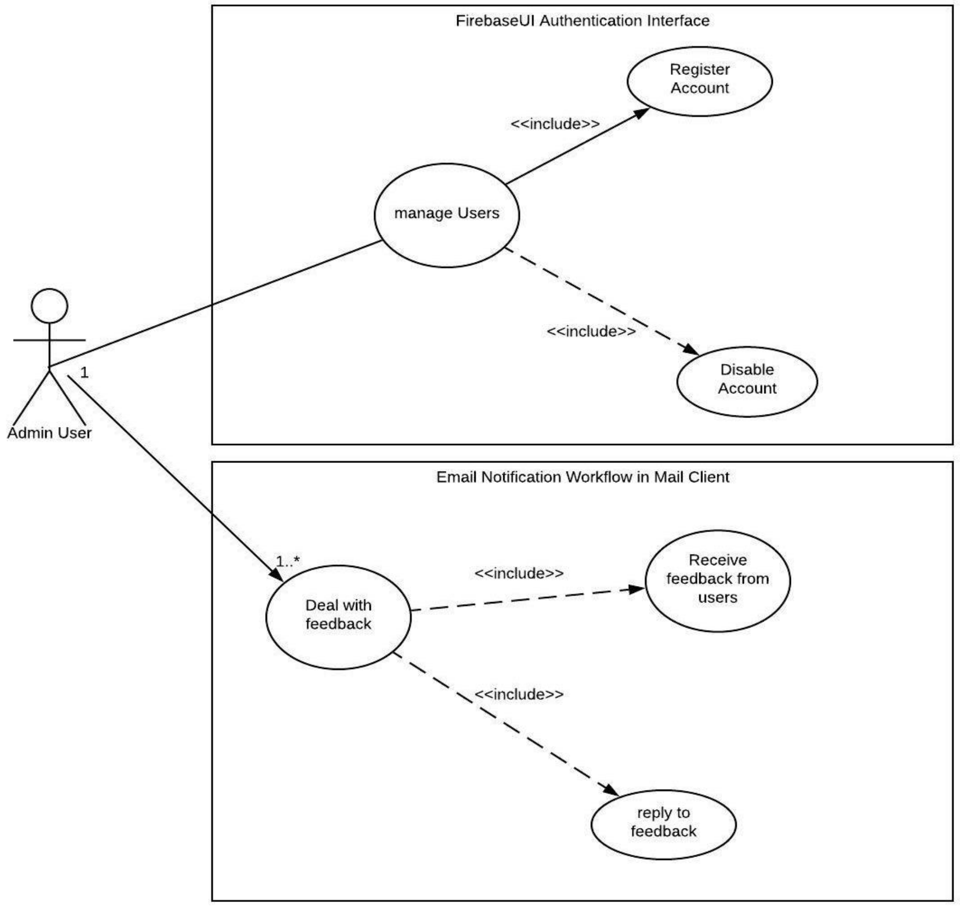
The Software Design Document is divided into 5 sections with various subsections.
The sections of the Software Design Document are:
1) Introduction,
2) Use Cases,
3) Design Overview,
4) System Object Model, and
5) Object Descriptions

2 Use Cases

2.1 Actors

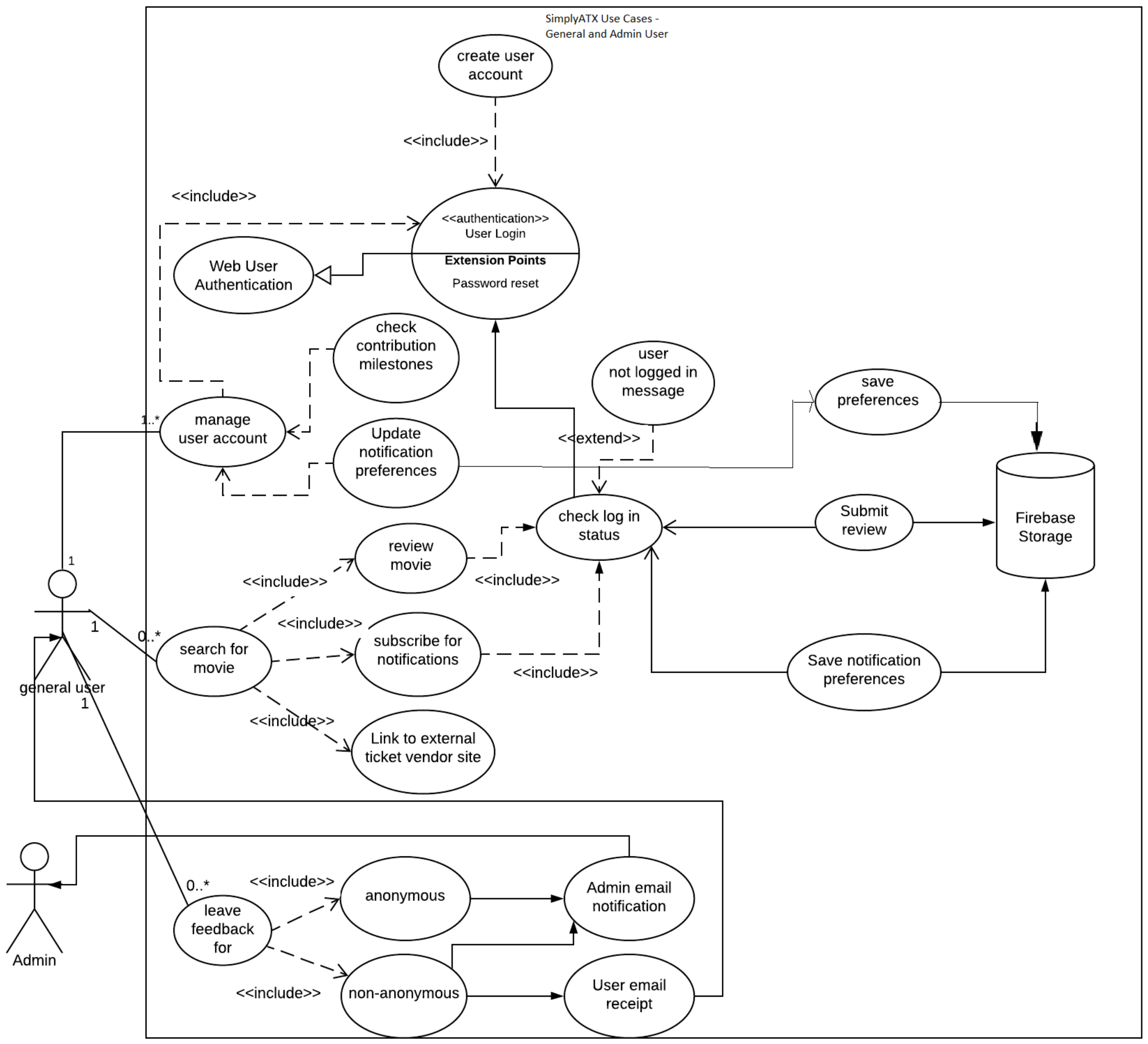
2.1.1 Administrative user

The Administrative User is a user who administers the system by overseeing accounts creation and system management. They also respond to user feedback if necessary.



2.1.2 General User with Admin for User Feedback

The general user can perform unauthenticated searches and be linked to external ticket providers. To leave a review for a title, sign up for notifications, or receive an activity milestone, the user must be authenticated.


SimplyATX Use Cases - General and Admin User
create user account
<<include>>
<<include>>
<<authentication>> User Login
Extension Points
Password reset
Web User Authentication
check contribution milestones
manage user account
1..*
Update notification preferences
user not logged in message
<<extend>>
save preferences
Firebase Storage
check log in status
Submit review
review movie
<<include>>
<<include>>
1
general user
1
1
0..*
search for movie
<<include>>
subscribe for notifications
<<include>>
Save notification preferences
<<include>>
Link to external ticket vendor site
Admin
0..*
<<include>>
anonymous
Admin email notification
leave feedback for
<<include>>
non-anonymous
User email receipt

2.2 Use Cases

2.2.1 Document Use Case - Login

Use case name: Web User Authentication		**ID:** 1	**Priority:** High
Primary actor: SimplyATX.com	**Source:** Admin, General User	**Use case type:** Business	**Level:** Detail
Interested Stakeholders:	Admin, General User		
Brief description:	This use case describes the account login process for a user. In this use case, the actor's goal is to provide a certifiable user name, and an email and password that the authentication system can verify.		
Goal:	● A successful account verification process.		
Success Measurement:	● Users are allowed access to their account on SimplyATX.com or denied access if invalid credentials are provided for an existing account.		
Precondition:	● User has an account already created and archived within the SimplyATX.com account management sub-system.		
Trigger:	● Users have accessed the SimplyATX.com website through a web browser tab in a new session.		
Relationships:	Include: GoogleAuthenticationProvider Email based Firebase-Authentication Extend: Depends on: Pre-existing email or Google Account		
Typical flow of events:	1. Users attempt to access SimplyATX.com website on a web browser. 2. Google DNS redirects user requests to the main screen. 3. Users can perform basic search functionality but must authenticate to leave reviews. 4. Site directs request to FirebaseUI Authentication system. 5. Authentication channels are queried: Google Auth or Email-Based Firebase Auth. 6. If passed, access it granted and site redirects to the User profile portal on the main site, SimplyATX.com. 7. If failed, the login page is updated with failure notification allowing a retry.		
Assumptions:	The user has an existing email or a Google account. The user has a working network and internet connection.		
Implementation Constraints and Specifications: If a user does not have an existing account, they are not prompted to create one. Simply attempting to log in and providing an email and password or signing in via an existing Google account will create a new account for the user. Firebase does this intentionally to make account creation simpler and drive increased conversion rates.			

2.2.2 Document Use Case - Leave Feedback

Use case name: Leave Feedback about the website		**ID:** 2	**Priority:** Low
Primary actor: SimplyATX.com	**Source:** General User	**Use case type:** Business	**Level:** Detail
Interested Stakeholders:	Admin, General User		
Brief description:	This use case describes how a general user may leave feedback, anonymously or not, from the site's main UI and receive a receipt email.		
Goal:	● Allow a general user to access the feedback interface to submit any information to the site's owners.		
Success Measurement:	● Users can access the feedback system and receive an acknowledgement email from SimplyATX.com upon submission.		
Precondition:	● Users need to know how to open the website		
Trigger:	● Users have accessed the SimplyATX.com website through a web browser tab in a new session.		
Relationships:	Include: Non- anonymous feedback Anonymous feedback Extend: Depends on: ZoHo Mail service API		
Typical flow of events:	1. Users attempt to access SimplyATX.com website on a web browser. 2. Google DNS redirects user requests to the main UI Login screen. 3. User clicks on Feedback link. 4. UI redirects User to Feedback form page. 5. User enters feedback into form fields and submits. 6. Data is sent to data storage and mail service is triggered. 7. Receipt email is sent to the user as confirmation.		
Assumptions:	The user has a working network. The user has explored the features on the website.		
Implementation Constraints and Specifications: Feedback form may require the addition of a security CAPTCHA to prevent bots or other malicious forms of behavior from breaking the systems or site.			

2.2.3 Document Use Case - Search

Use case name: Search for a Title		**ID:** 3	**Priority:** High
Primary actor: SimplyATX.com	**Source:** General User	**Use case type:** Business	**Level:** Detail
Interested Stakeholders:	General User		
Brief description:	This use case describes how a general user may search for title information by a keyword.		
Goal:	● Allow a general user to operate a search interface to exploit the search API available through the website.		
Success Measurement:	● The search API returns a collection of results, either full or empty, in no particular order.		
Precondition:	● Users know how to open the website and enters a search		
Trigger:	● The main interface preloads with the search bar after a user navigates to the site.		
Relationships:	Include: IMDB.com AFI.com Extend: Depends on: IMDB.com - open catalog access AFI.com - open catalog access		
Typical flow of events:	1. From the main page, a user enters a search term and clicks the search button. 2. The request is sent to the WebService API of SimplyATX.com. 3. That service calls the AFI and IMDB search APIs and waits for a result set. 4. That result set is then processed by the service and returns the conditioned set to the main interface. 5. The main page builds a wrapper set around the results and displays these as a list of links, one per title. 6. The user then selects a desired title by clicking on the displayed result. 7. This event triggers a service call to the SimplyATX.com API. 8. The service makes an internal request to IMDB for specific movie information. 9. The resultant catalog information is then processed by the service which conditions it and returns it to the main site. 10. The main site parses the information and displays it for the user.		
Assumptions:	The user has a working network.		
Implementation Constraints and Specifications: All information is based on external sites and resources; no movie or TV listing is sourced by SimplyATX.com. Some title information may not be available via the AFI or IMDB external datasets.			

2.2.4 Document Use Case - Leave Review

Use case name: Leave Review for a movie		**ID:** 4	**Priority:** High
Primary actor: SimplyATX.com	**Source:** General User	**Use case type:** Business	**Level:** Detail
Interested Stakeholders:	General User		
Brief description:	This use case describes how an existing user may leave a review for a movie or tv show.		
Goal:	● Allow an existing user to click on an emoji to leave a simple but unique and intuitive title review.		
Success Measurement:	● Users are able to leave review for movies after they log into the website		
Precondition:	● Users must log into their account. ● Users must query a specific Movie listing.		
Trigger:	● After a successful query through the SimplyATX.com Search API, a selected title will present an option for leaving a review.		
Relationships:	Include: IMDB.com AFI.com Extend: SimplyATX.com user resources Depends on: IMDB.com - open catalog access AFI.com - open catalog access		
Typical flow of events:	1. Users must log into their account. 2. In the successful search results a user must select a listing of Movies by title. 3. The main interface makes a call to the SImplyATX.com search service API. 4. The service makes a specific request to IMDB for the title's detailed information. 5. The service conditions the dataset and returns it to the main interface. 6. The webpage receives the data and parses it for display, adding a link for a review option on SimplyATX.com. 7. The user clicks the link for reviewing and is presented with a modal window view interface. 8. The interface contains a template for name, email, subject and message.		
Assumptions:	The user has a working network. The user is logged into SimplyATX.com. The user has searched for a TV show or Movie listing.		
Implementation Constraints and Specifications: All review information is non-verbal and uses proprietary review feedback content for ease of tracking and reuse management.			

2.2.5 Document Use Case - Subscribe to Notifications

Use case name: Subscribe to notifications		**ID:** 5	**Priority:** High
Primary actor: SimplyATX.com	**Source:** General User	**Use case type:** Business	**Level:** Overview
Interested Stakeholders:	General User		
Brief description:	This use case describes how a general user may subscribe to notifications concerning a specific predefined category of specialty showings.		
Goal:	• Allow a general user to subscribe to notification messages from SimplyATX.com concerning a specific category of TV show or Movie.		
Success Measurement:	• Users have access to the notification system through the main interface on SimplyATX.com.		
Precondition:	• Users are logged in to SimplyATX.com.		
Trigger:	•		
Relationships:	Include: ZoHo mail Extend: SimplyATX.com, user account settings Depends on: ZoHo mail service API		
Typical flow of events:	1. From the main interface a user selects a link to access their personal account information. 2. The main interface displays, from the main site, a UI that contains the users account info, including a notification option. 3. The user selects this option to display a modal interface view that presents a categorical selection list and an update button. 4. This makes a request of the SimplyATX.com service API to update the users account in Firebase cloud storage. 5. Upon SimplyATX.com receiving updates in the selected category for upcoming films or reviews, the user accounts which signed up for the request notification are sent messages via the ZoHo mail SMTP server.		
Assumptions:	ZOHO mail is operational and users email address is correct The user has a working network. The user is logged into SimplyATX.com.		
Implementation Constraints and Specifications: Any and all categorical notification assignments are limited to any generic genre available from external media sources.			

2.2.6 Document Use Case - Activity Milestone

Use case name: Activity Milestone		**ID:** 6	**Priority:** High
Primary actor: SimplyATX.com	**Source:** General User	**Use case type:** Business	**Level:** Overview
Interested Stakeholders:	General User		
Brief description:	This use case describes how a general user achieves and receives Activity Milestone awards within SimplyATX.com award system.		
Goal:	● Allow a user to earn and add a badge icon to their account profile through the SimplyATX.com award system.		
Success Measurement:	● Users may complete pre-specified metrics tracked through SimplyATX.com that fulfill the criteria programmed into the award system. Upon meeting the requirement, the user is awarded with the specific milestone achievement.		
Precondition:	● User has an account with SimplyATX.com and has completed the required activities on SimplyATX.com.		
Trigger:	● The User satisfies the programmed metric.		
Relationships:	Include: SimplyATX.com - features and subsystems Extend: SimplyATX.com user account management Depends on: SimplyATX.com - award subsystem		
Typical flow of events:	1. **Pioneer**: Leave 1st custom review 2. **Well-informed**: Subscribe to notifications 3. **Grrr!**: Leave 5 “angry” reviews 4. **I’m not crying, you’re crying**: Leave 5 “sad” reviews 5. **Heh…**: Leave 5 “funny” reviews 6. **Is the paint dry?**: Leave 5 “boring” reviews 7. **Much ado about nothin’**: Leave 5 “meh” reviews 8. **Huh?**: Leave 5 “hmm” reviews 9. **WTF?!**: Leave 10 “huh” reviews 10. **Bruh**: Leave 5 “expressionless” reviews		
Assumptions:	The user has a working network. The user is logged into SimplyATX.com. The user has been leaving reviews for movies		
Implementation Constraints and Specifications: Content and badges does not violate any trademark, logo nor copyrighted media. All media used in this system was obtained from open license media providers.			

3 Design Overview

3.1 Introduction

The Design Overview is a section to introduce and give a brief overview of the design. The System Architecture is a way to give the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient themselves to the design and see a summary before proceeding into the details of the design.

3.2 System Architecture

Refer to **attached** architecture diagram; zoom in and scroll as needed for visibility.
Diagram is included as an attachment to preserve readability for connectors between components.

TAD_SimplyATX.pdf

Observe the numbered gray triangle, normally on the bottom right of a component when possible: △10
This represents the component's number and is used to describe the sequence of execution.
Example: Let us examine the "Leave a review" use case sequence:
(perform search sequence mandatory) + 10, 15, 10, 9

What this tells us is that we have to first perform the sequence of execution for performing a search - it is a mandatory prerequisite for leaving a review for any given title. Next, we start from the last number of the search sequence, which is 9, the UI. We go to 10, the back-end API. Important note: we procure user data directly via client-side code but we make a back-end call to the API for any processing and to invoke business logic as needed. Then we visit Firebase Storage at 15, to save the new review. We get a return status in our API and get back to the UI, informing the user that their actions have been accepted and saved.

The diagram consists of seven tiers, with flow of execution between the tiers and components therein clearly outlined. Refer to the bottom left chart for the line color legend. All ports and types of traffic are annotated where possible. The seven tiers are: client, access, web, application, managed services, data, and storage. The dashed yellow line around components 9 and 10 indicates that they both reside as Google Cloud Platform projects. Memory, storage size, and CPU count or speed are also annotated where possible. For component 10, the 1000m CPU speed is due to the application residing in a shared tenant environment at the free tier of service. This means it receives only a share of a single CPU's time.

4 System Object Model

4.1 Introduction

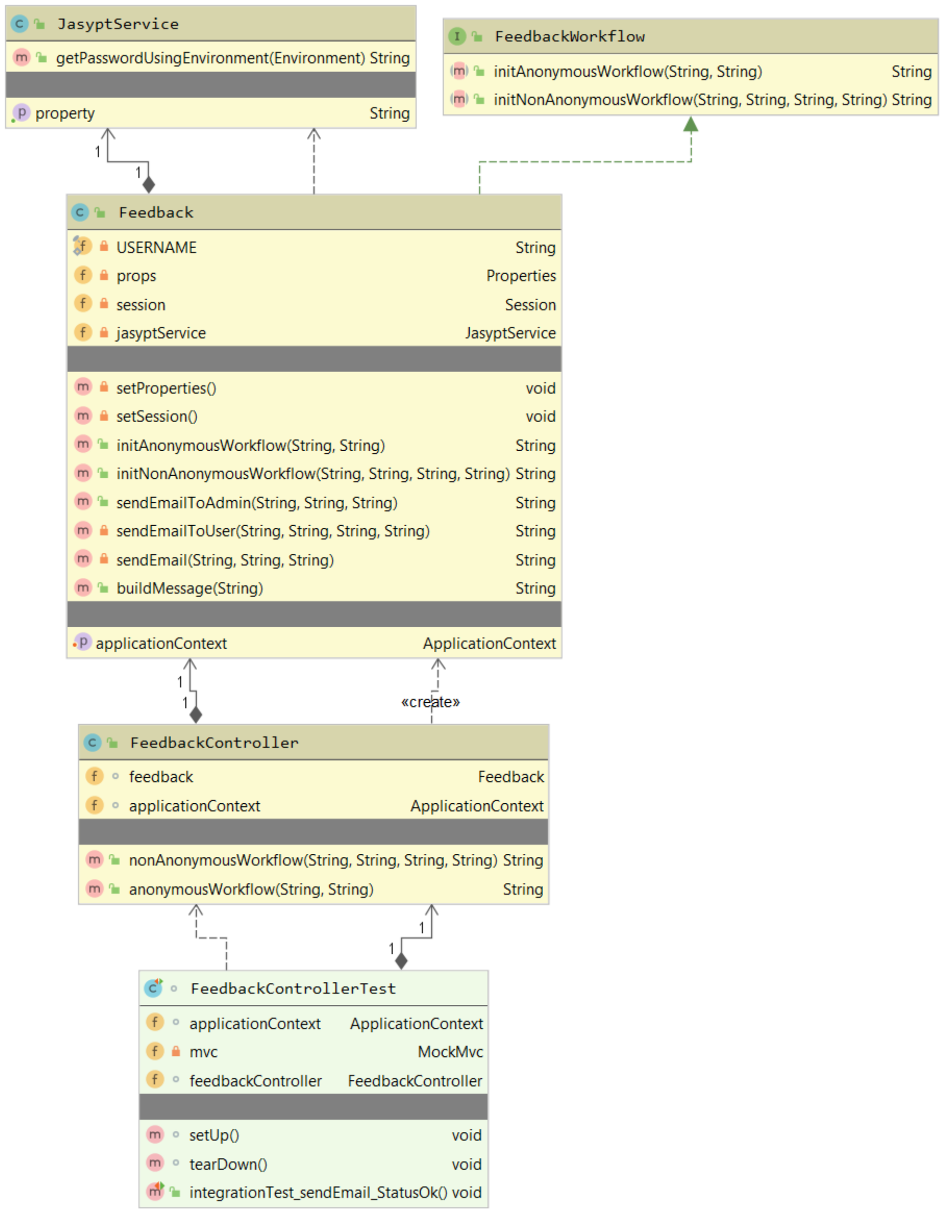
The System Object Model Section allows for a description of the subsystems in use. This allows for describing the system in an overall manner to show the different groupings of parts into respective systems.

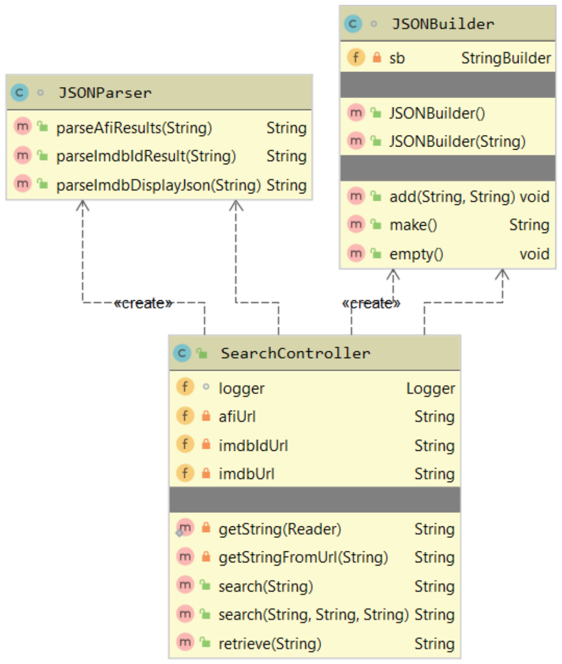
4.2 Subsystems

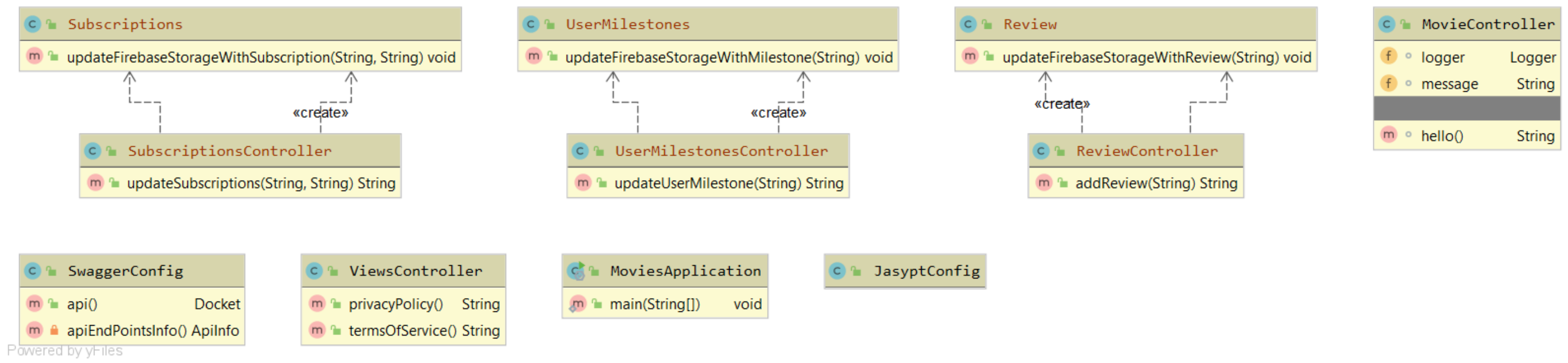
Name	Interface	Description
AFI	*catalog.afi.com*	Used the Search API to accrue all intermediate search results for movies over the last 100 years.
IMDB	*V2.sg.media-imdb.com*	Used to locate proprietary movie ID information based on title.
	m.imdb.com	Used to obtain specific details covering numerous characteristics of a title's relevant media and metadata.
ZoHo Mail	*https://www.zoho.com/mail/*	Using the online email provider for all messaging between SimplyATX.com and its visitors or users.
Firebase	*https://firebase.google.com/*	Used for user authentication and persistent storage of user data (e.g., reviews, milestones)

5 Object Descriptions

5.1 Class diagram


JasyptService
getPasswordUsingEnvironment(Environment) String
property String
FeedbackWorkflow
initAnonymousWorkflow(String, String) String
initNonAnonymousWorkflow(String, String, String, String) String
1
1
Feedback
USERNAME String
props Properties
session Session
jasyptService JasyptService
setProperties() void
setSession() void
initAnonymousWorkflow(String, String) String
initNonAnonymousWorkflow(String, String, String, String) String
sendEmailToAdmin(String, String, String) String
sendEmailToUser(String, String, String, String) String
sendEmail(String, String, String) String
buildMessage(String) String
applicationContext ApplicationContext
1
1
«create»
FeedbackController
feedback Feedback
applicationContext ApplicationContext
nonAnonymousWorkflow(String, String, String, String) String
anonymousWorkflow(String, String) String
1
1
FeedbackControllerTest
applicationContext ApplicationContext
mvc MockMvc
feedbackController FeedbackController
setUp() void
tearDown() void
integrationTest_sendEmail_StatusOk() void


JSONBuilder
sb StringBuilder
JSONBuilder()
JSONBuilder(String)
add(String, String) void
make() String
empty() void
JSONParser
parseAfiResults(String) String
parseImdbIdResult(String) String
parseImdbDisplayJson(String) String
«create»
«create»
SearchController
logger Logger
afiUrl String
imdbIdUrl String
imdbUrl String
getString(Reader) String
getStringFromUrl(String) String
search(String) String
search(String, String, String) String
retrieve(String) String



The @RequestMapping annotation maps controllers to all requests starting with "/{url_path}". Methods in a controller need only declare paths relative to that. The @Controller annotation indicates that a particular class serves the role of a *controller*.

Table 2: Class description at the feature level

Class / Classes	Description
MoviesApplication	This class is annotated as @SpringBootApplication. It serves as the entry point into the application and
FeedbackController, Feedback	The FeedbackController class is responsible for processing feedback requests issued through the UI by visitors to SimplyATX.com. Feedback is instantiated by FeedbackController and performs the business logic. For example, the buildMessage(String name) method will consume a user's name as a string and return a pre-defined email message body containing a custom greeting using the user's name.
SearchController	The SearchController class is responsible for processing search requests issued by users through the UI on SimplyATX.com.
SearchController.JSONBuilder	The JSONBuilder is a wrapper for the internal Java.String.stringbuilder class with added methods to encapsulate java strings as json strings for packet transmission via service request replies of the SearchController exposed methods.
SearchController.JSONParser	The JSONParser is a custom class designed for parsing the received data packets from both IMDB and AFI API calls.

ReviewController	The ReviewController class is responsible for processing and managing the review system within SimplyATX.com for users that create review objects through the main UI at the site level.
MovieController	The MovieController class is responsible for processing movie and or TV show title analysis requests by users. It is a planned class that will direct the flow of logic for more detailed title information and possible analytics. It is also responsible for providing a basic REST endpoint for application keep-alive monitoring, to ensure the availability of the API.
UserMilestoneController	The UserMilestoneController class is responsible for managing the milestone system within SimplyATX.com and tracking users' actions through the main UI at the site level.
SubscriptionsController	The SubscriptionsController class is responsible for managing the subscription system within SimplyATX.com and tracking user subscriptions based on ongoing activity outside of the main site.
JasyptConfig, JasyptService	These classes are used to procure sensitive application data, such as passwords. The passwords are encrypted and stored in the application properties file and are decrypted at runtime when needed.
SwaggerConfig	This is a helper class that provides a detailed description of the entire API - all available HTTP requests are evaluated by this library and is accessible via swagger-ui.html, a dynamically created UI for documenting the API endpoints.
ViewsController	This controller returns views, which are maintained as static HTML in the backend. Currently it returns the site's privacy and terms of service statements.

5.2 Description of Attributes and Methods (Client-side code)

This section will cover what is not easily described by a class diagram. The SimplyATX website contains much of its logic as client-side JavaScript code.

app.js:
- firebaseConfig: JSON object which registers our application with Firebase
- uiConfig: this object configures how we use the drop-in FirebaseUI authentication and defines which authentication methods we make available to the user - email and Google provider sign-in.
- initApp(): function to initialize the application with the defined Firebase services and get the initial user authentication state.
- logout(): allows the user to log out if they are signed in.

feedbackRestUtility.js: contains logic to perform HTTP GET requests to the API for handling user feedback.

- anonymousFeedback(subject, message): makes an HTTP GET request to the API without a username or email.
- nonAnonymousFeedback(name, email, message): makes an HTTP GET request with username and email. User will receive a receipt at the end of this workflow.

searchRestUtility.js: contains logic to perform HTTP GET requests to the API for obtaining JSON data from AFI and IMDB regarding the user's search criteria.

- searchImdb(origin, searchCriteria): Will obtain IMDB JSON payload for parsing and displaying to the user.
- searchAfi(origin, searchCriteria): Will obtain AFI JSON payload for parsing and displaying to the user.

show.js: defines what comprises a "show" and it's associated attributes, such as title, year of release, description, and a URL for the poster. It will consume these as parameters passed to it and return back an HTML element with the title's provided details.

5.2.1 User Login

Sequence(1, 6, 7, 8, 9, 3, 4 OR 5, 3, 9)

A User connects through local DNS with Google DNS->Google GTM->GoogleLTM->SimplyATX.com
At the login screen the user inputs their login credentials which are validated by FirebaseUI Auth->Google/Firebase Authentication-> then returns to SimplyATX.com with the login confirmation or failure result.

5.2.2 Perform Search

Sequence((login sequence optional) + 1, 6, 7, 8, 9, 10, 13 AND 14, 10, 9)

From the main UI of SimplyATX.com a user/visitor may activate the 'Search' feature directly this makes a service request on SimplyATX.com Web Service API->AFI/Search API which returns a list of possible results, and then upon a user's selection of a title for more information a request is issued to the IMDB.com API, all requests return payloads to the main UI on SimplyATX.com

5.2.3 Leave a Review

Sequence(perform search sequence mandatory) + 10, 15, 10, 9)

After a search request has been successfully issued and received by the user, when the review feature is selected a review form is opened in the main site allowing a user to make a review submission this makes a request on the SimplyATX.com Web Service API->Firebase Cloud Storage to add a record for the given user on the requested title->an acknowledgement is then posted on the main UI of SimplyATX.com returning to the search results screen from where the review request was originated.

5.2.4 Subscribe to notification

Sequence((login sequence mandatory) + 10, 15, 10, 11, 12, 1)
A User connects through local DNS with Google DNS->Google GTM->GoogleLTM->SimplyATX.com
At the login screen the user inputs their login credentials which are validated by FirebaseUI Auth->Google/Firebase Authentication-> then returns to SimplyATX.com with the login confirmation or failure result.
After users have successfully logged in, users can click on which content to subscribe to, those settings will be saved to Firebase Cloud Storage through SimpleATX Web Service API. Important note: we procure user data directly via client-side code but we make a back-end call to the API for any processing and to invoke business logic as needed. Then we visit Firebase Storage at 15, to save the new settings. We get a return status in our API and get back to the UI, if there is an update on the content of subscription, users will be informed.

5.2.5 Leave Feedback

Sequence (1, 6, 7, 8, 9, 10 , 11, 12, 1, AND 2)
User connects thru local DNS with Google DNS->Google GTM->GoogleLTM->SimplyATX.com
A feedback form will pop up for users to provide feedback. Users have two choices: either leave anonymous feedback without entering an email address, or leave non-anonymous feedback by entering an email address.
After the user hits the submit button, SimpleATX Web Service API will connect to ZOHO Mail Server. From here, emails will be sent to the admin email account. If this is non-anonymous feedback, another email will be sent to the user to confirm the admin has received the feedback.

5.2.6 Activity Milestone
Sequence((login sequence mandatory) + (search sequence mandatory) + (review sequence mandatory) + 10, 15 10, 11, 12, 1)
Upon successful login, search and review procedures are completed the MilestoneActivity system will actively monitor all transactions and in the event of a completed milestone metric the SimplyATX.com Web Service API will make an update on the users account through the Firebase Cloud Storage and then a call to the Zoho mail service API to notify the user of a completed milestone.

5.2 User Interface Mapping Diagram

See attached PDF export:

UI_Map_SimplyATX.pdf