Catch Movie!

SSP8 - Group 4

Casuarina, Dobrasiewicz Piotr, Godeliva Petrina M, Tan Jian Wei, Zhang Yuhan

Agenda

- Introduction to Catch Movie
- Requirements Elicitation & Analysis
- Conceptual Models: Architecture & Class Design
- Dynamic Models: Sequence Diagram & Dialog Map
- Testing: Black box & White box
- Application Live Demo
- Learning Points & Challenges
- Possible Future Extensions

Introduction to Catch Movie

About Catch Movie

Provide a **unified platform** for users to **view movie information and book movie tickets from various cinemas**: Cathay, Golden Village and Shaw.

Value Proposition

- **Quick search and comparison** of movie schedules in one app
- **Eliminate trouble of opening multiple** cinema websites
- Convenient way to **find showtime that best suits our needs from today to next 4 days**

Product Scope & Targeted Audience

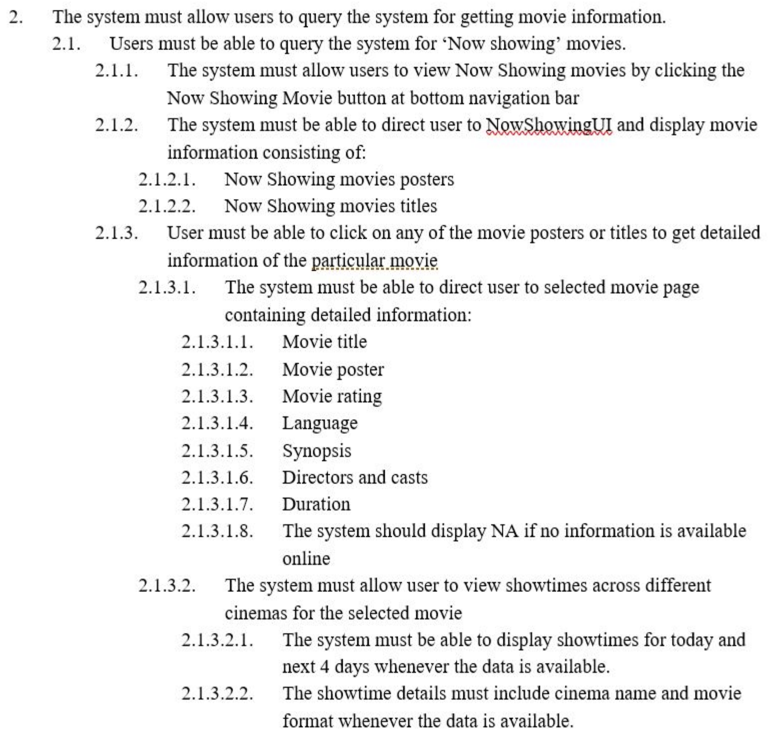
The application is mainly targeted for people in Singapore as it utilizes publicly available movie data in Singapore only

Product Functions

- **Login** via gmail account
- Retrieving movie information of **Now Showing and Upcoming movies compiled across different cinemas** in Singapore:
    - **Movie details** (Synopsis, rating, casts etc)
    - **Showtimes** (Today to next 4 days)
- **Direct search** of movie information by providing **movie title and movie date as input**
- **Direct booking of movie tickets** via link to official cinema booking page
- Retrieving **ongoing promotions across different cinemas** in Singapore
- Viewing **support/basic FAQ** regarding how to use the application
- **Submitting rating** for the application

Requirements Elicitation & Analysis

Requirements Design



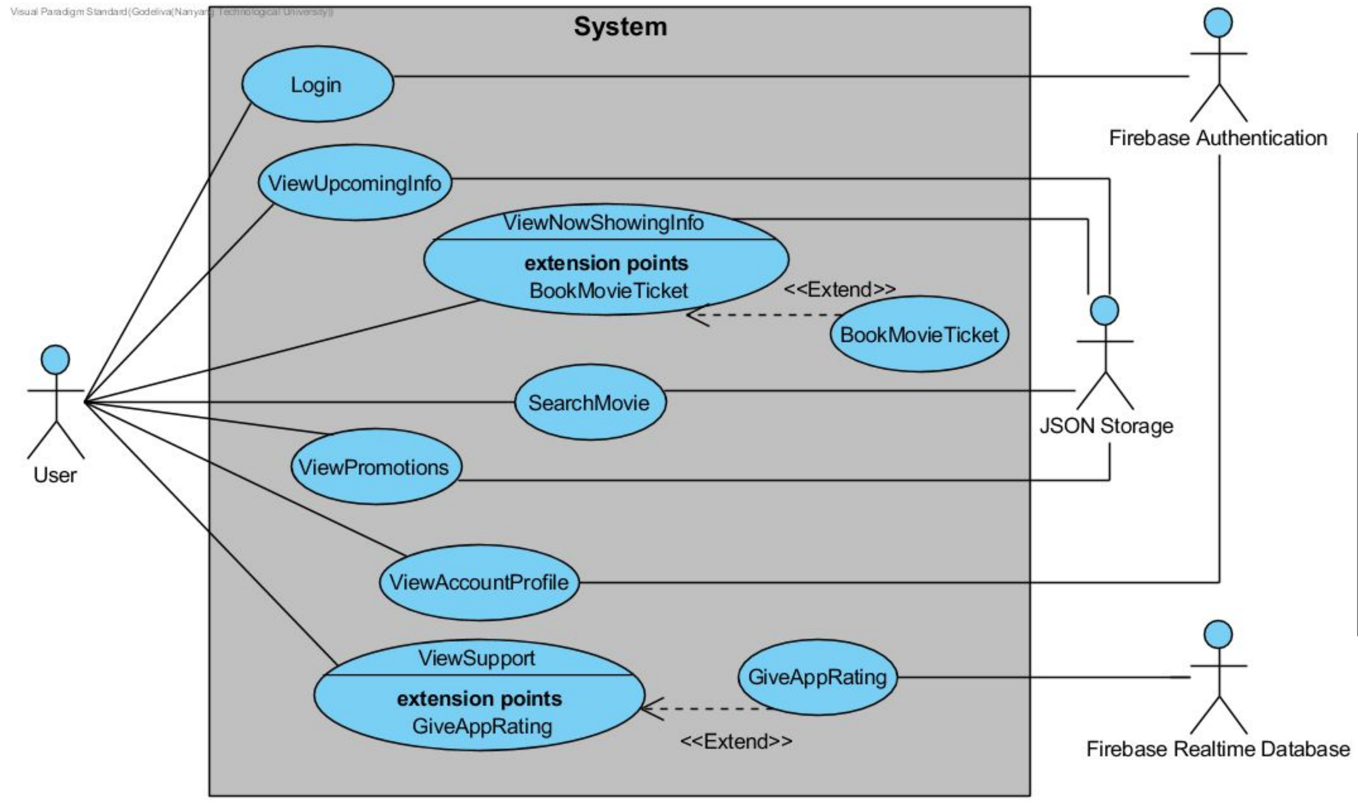
2. The system must allow users to query the system for getting movie information.
   - 2.1. Users must be able to query the system for 'Now showing' movies.
     - 2.1.1. The system must allow users to view Now Showing movies by clicking the Now Showing Movie button at bottom navigation bar
     - 2.1.2. The system must be able to direct user to NowShowingUI and display movie information consisting of:
       - 2.1.2.1. Now Showing movies posters
       - 2.1.2.2. Now Showing movies titles
     - 2.1.3. User must be able to click on any of the movie posters or titles to get detailed information of the particular movie
       - 2.1.3.1. The system must be able to direct user to selected movie page containing detailed information:
         - 2.1.3.1.1. Movie title
         - 2.1.3.1.2. Movie poster
         - 2.1.3.1.3. Movie rating
         - 2.1.3.1.4. Language
         - 2.1.3.1.5. Synopsis
         - 2.1.3.1.6. Directors and casts
         - 2.1.3.1.7. Duration
         - 2.1.3.1.8. The system should display NA if no information is available online
       - 2.1.3.2. The system must allow user to view showtimes across different cinemas for the selected movie
         - 2.1.3.2.1. The system must be able to display showtimes for today and next 4 days whenever the data is available.
         - 2.1.3.2.2. The showtime details must include cinema name and movie format whenever the data is available.

Take *ViewNowShowing* use case as example

- **Atomized** requirements
- Logically **structured** requirements

→ **Easily tested & verifiable** requirements

Use Case Diagram



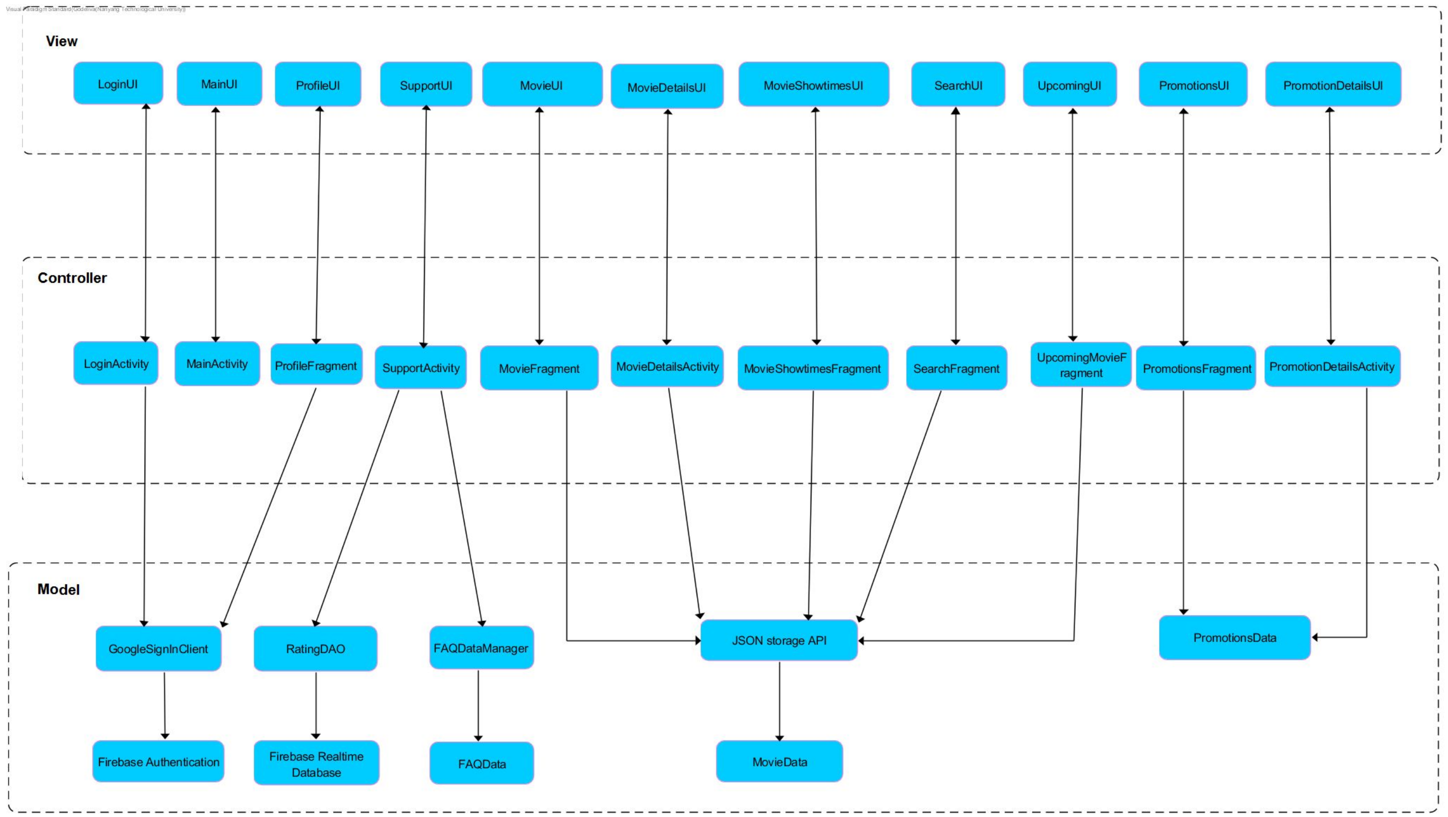
Initiating actor:
User

Participating actor:

- *Firebase Authentication: Authenticates user account via gmail*
- *JSON Storage: Stores web scraping data*
- *Firebase Realtime Database: Stores users' submitted ratings*

Conceptual Models

Architecture Design: MVC



Defines presentation to user

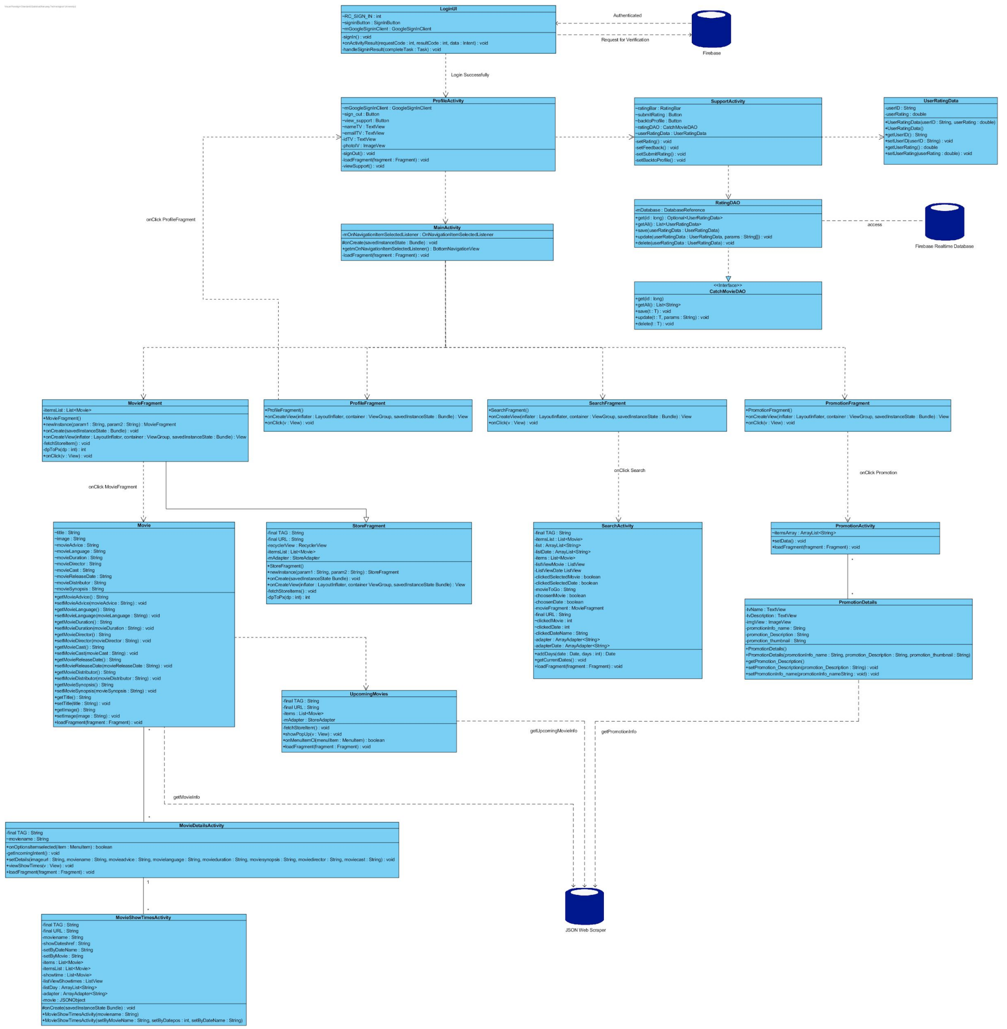
Manages user interaction and captures user input

Contains data involved

Architecture Design Benefits

- Separation of Model from View and Controller → allows **multiple views of same model**
  - Ex: Now Showing UI, Upcoming UI, Search UI all use same model but different presentation
- Views are **not tightly coupled** with core functionalities
  - Can **easily change UI design**
- **High cohesion** due to **logical grouping of related actions** together
  - Each View has its own Controller and grouped together with its specific model

Class Diagram

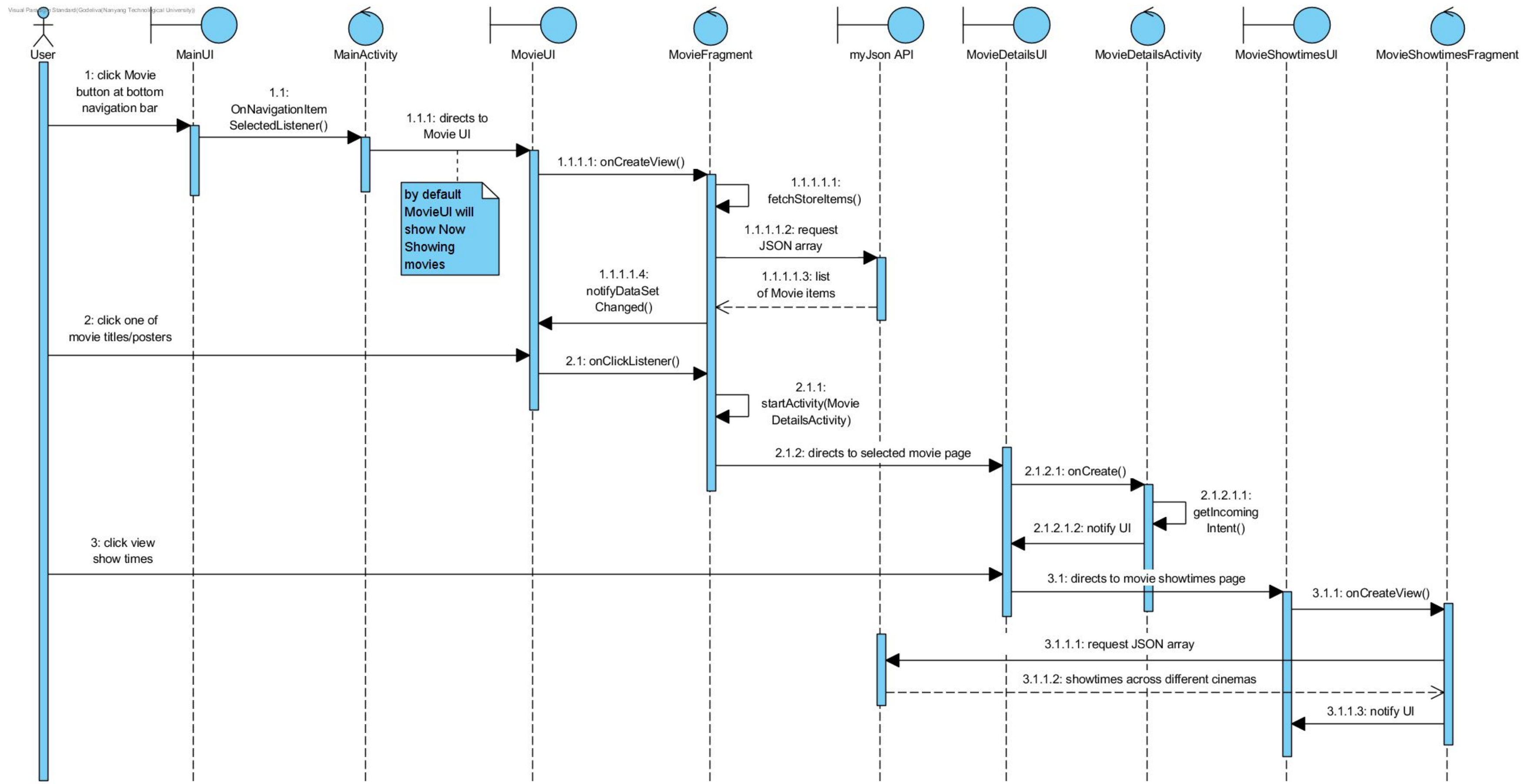

LoginUI
~RC_SIGN_IN : int
~signinButton : SignInButton
~mGoogleSigninClient : GoogleSignInClient
-signIn() : void
+onActivityResult(requestCode : int, resultCode : int, data : Intent) : void
-handleSigninResult(completeTask : Task) : void
Authenticated
Request for Verification
Firebase
Login Successfully
ProfileActivity
~mGoogleSignInClient : GoogleSignInClient
~sign_out : Button
~view_support : Button
~nameTV : TextView
~emailTV : TextView
-idTV : TextView
-photoIV : ImageVew
-signOut() : void
-loadFragment(fragment : Fragment) : void
-viewSupport() : void
SupportActivity
~ratingBar : RatingBar
~submitRating : Button
~backtoProfile : Button
~ratingDAO : CatchMovieDAO
~userRatingData : UserRatingData
-setRating() : void
-setFeedback() : void
-setSubmitRating() : void
-setBacktoProfile() : void
UserRatingData
-userID : String
-userRating : double
+UserRatingData(userID : String, userRating : double)
+UserRatingData()
+getUserID() : String
+setUserID(userID : String) : void
+getUserRating() : double
+setUserRating(userRating : double) : void
RatingDAO
-mDatabase : DatabaseReference
+get(id : long) : Optional<UserRatingData>
+getAll() : List<UserRatingData>
+save(userRatingData : UserRatingData)
+update(userRatingData : UserRatingData, params : String[]) : void
+delete(userRatingData : UserRatingData) : void
access
Firebase Realtime Database
<<Interface>>
CatchMovieDAO
+get(id : long)
+getAll() : List<String>
+save(t : T) : void
+update(t : T, params : String) : void
+delete(t : T) : void
onClick ProfileFragment
MainActivity
-mOnNavigationItemSelectedListener : OnNavigationItemSelectedListener
#onCreate(savedInstanceState : Bundle) : void
+getmOnNavigationItemSelectedListener() : BottomNavigationView
-loadFragment(fragment : Fragment) : void
MovieFragment
-itemsList : List<Movie>
+MovieFragment()
+newInstance(param1 : String, param2 : String) : MovieFragment
+onCreate(savedInstanceState : Bundle) : void
+onCreateView(inflater : LayoutInflator, container : ViewGroup, savedInstanceState : Bundle) : View
-fetchStoreItem() : void
-dpToPx(dp : int) : int
+onClick(v : View) : void
ProfileFragment
+ProfileFragment()
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onClick(v : View) : void
SearchFragment
+SearchFragment()
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onClick(v : View) : void
PromotionFragment
+PromotionFragment()
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onClick(v : View) : void
onClick MovieFragment
onClick Search
onClick Promotion
Movie
~title : String
~image : String
~movieAdvice : String
~movieLanguage : String
~movieDuration : String
~movieDirector : String
~movieCast : String
~movieReleaseDate : String
~movieDistributor : String
~movieSynopsis : String
+getMovieAdvice() : String
+setMovieAdvice(movieAdvice : String) : void
+getMovieLanguage() : String
+setMovieLanguage(movieLanguage : String) : void
+getMovieDuration() : String
+setMovieDuration(movieDuration : String) : void
+getMovieDirector() : String
+setMovieDirector(movieDirector : String) : void
+getMovieCast() : String
+setMovieCast(movieCast : String) : void
+getMovieReleaseDate() : String
+setMovieReleaseDate(movieReleaseDate : String) : void
+getMovieDistributor() : String
+setMovieDistributor(movieDistributor : String) : void
+getMovieSynopsis() : String
+setMovieSynopsis(movieSynopsis : String) : void
+getTitle() : String
+setTitle(title : String) : void
+getImage() : String
+setImage(image : String) : void
+loadFragment(fragment : Fragment) : void
StoreFragment
-final TAG : String
-final URL : String
-recyclerView : RecyclerView
-itemsList : List<Movie>
-mAdapter : StoreAdapter
+StoreFragment()
+newInstance(param1 : String, param2 : String) : StoreFragment
+onCreate(savedInstanceState Bundle) : void
+onCreateView(inflater : LayoutInflater, container ViewGroup, savedInstanceState Bundle) : View
-fetchStoreItems() : void
-dpToPx(dp : int) : int
SearchActivity
-final TAG : String
-itemsList : List<Movie>
-list : ArrayList<String>
-listDate : ArrayList<String>
-items : List<Movie>
-listViewMovie : ListView
-ListViewDate ListView
-clickedSelectedMovie : boolean
-clickedSelectedDate : boolean
-movieToGo : String
-choosenMovie : boolean
-choosenDate : boolean
-movieFragment : MovieFragment
-final URL : String
~clickedMovie : int
~clickedDate : int
-clickedDateName : String
-adapter : ArrayAdapter<String>
-adapterDate : ArrayAdapter<String>
+addDays(date : Date, days : int) : Date
+getCurrentDates() : void
+loadFragment(fragment : Fragment) : void
PromotionActivity
~itemsArray : ArrayList<String>
+setData() : void
+loadFragment(fragment : Fragment) : void
*
*
PromotionDetails
-tvName : TextView
-tvDescription : TextView
-imgView : ImageView
-promotionInfo_name : String
-promotion_Description : String
-promotion_thumbnail : String
+PromotionDetails()
+PromotionDetails(promotionInfo_name : String, promotion_Description : String, promotion_thumbnail : String)
+getPromotion_Description()
+setPromotion_Description(promotion_Description : String) : void
+setPromotionInfo_name(promotionInfo_nameString : void) : void
UpcomingMovies
-final TAG : String
-final URL : String
-items : List<Movie>
-mAdapter : StoreAdapter
-fetchStoreItem() : void
+showPopUp(v : View) : void
+onMenuItemCl(menuItem : MenuItem) : boolean
+loadFragment(fragment : Fragment) : void
getUpcomingMovieInfo
getPromotionInfo
*
getMovieInfo
*
MovieDetailsActivity
-final TAG : String
~moviename : String
+onOptionsItemselected(item : MenuItem) : boolean
-getIncomingIntent() : void
+setDetails(imageurl : String, moviename : String, movieadvice : String, movielanguage : String, movieduration : String, moviesynopsis : String, moviedirector : String, moviecast : String) : void
+viewShowTimes(v : View) : void
+loadFragment(fragment : Fragment) : void
1
*
MovieShowTimesActivity
-final TAG : String
-final URL : String
-moviename : String
-showDateshref : String
-setByDateName : String
-setByMovie : String
-items : List<Movie>
-itemsList : List<Movie>
-showtime : List<Movie>
-listViewShowtimes : ListView
-listDay : ArrayList<String>
-adapter : ArrayAdapter<String>
-movie : JSONObject
#onCreate(savedInstanceState Bundle) : void
+MovieShowTimesActivity(moviename : String)
+MovieShowTimesActivity(setByMovieName : String, setByDatepos : int, setByDateName : String)
JSON Web Scraper

Class Design Considerations

- **Single Responsibility** → One class one responsibility/functionalities
    - Example:
- **Open-closed principle** → Open for extension but closed for modification
    - Example: Parent class StoreFragment with sub-class MovieFragment. StoreFragment is closed for modification but is open for extension by new subclasses in the future
- Implement **DAO pattern** → hide from the application logic the complexities involved in performing operations to database
    - Example: RatingDAO implements CatchMovieDAO to store ratings submitted by users to Firebase
    - Easy to add new database implementation (MySQL etc) for new DAO in the future

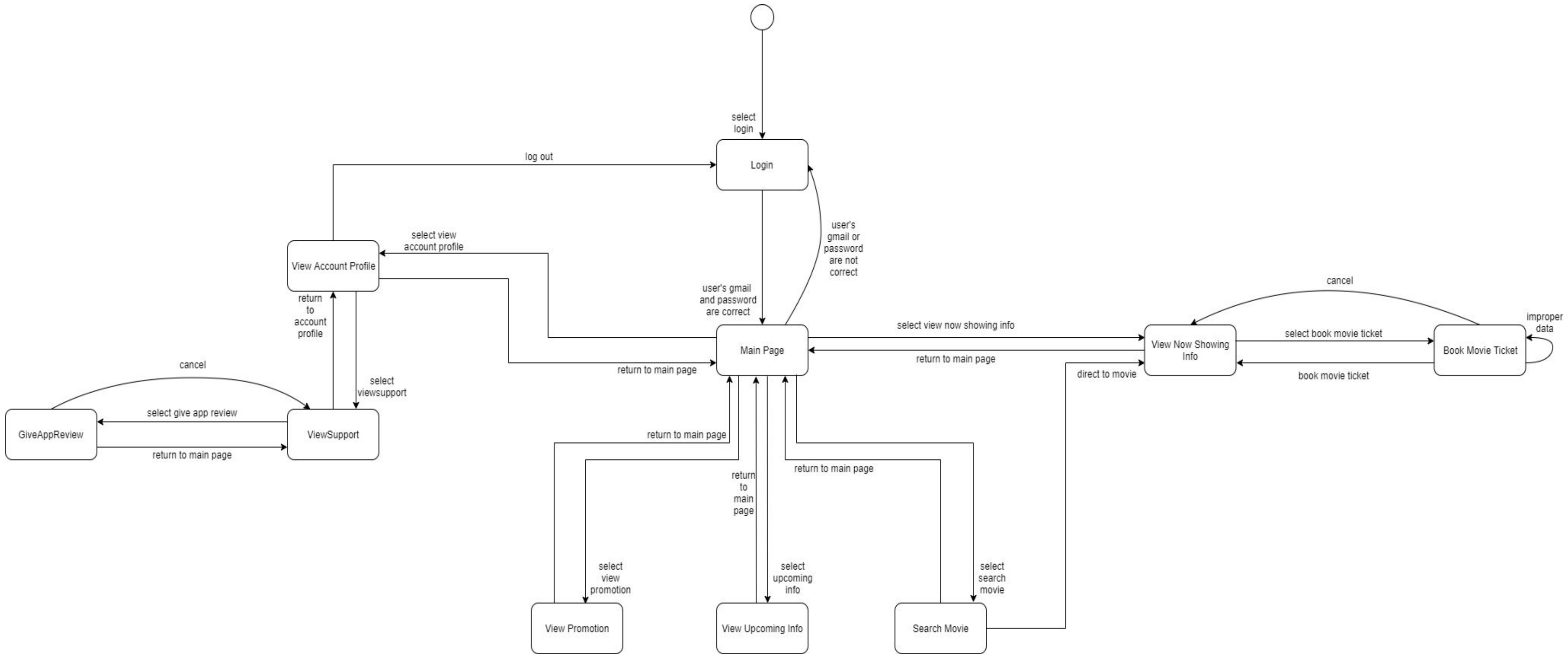
Dynamic Models

Sequence Diagram: *ViewNowShowingInfo*



Visual Paradigm Standard(Godeliva(Nanyang Technological University))
User
MainUI
MainActivity
MovieUI
MovieFragment
myJson API
MovieDetailsUI
MovieDetailsActivity
MovieShowtimesUI
MovieShowtimesFragment
1: click Movie button at bottom navigation bar
1.1: OnNavigationItem SelectedListener()
1.1.1: directs to Movie UI
by default MovieUI will show Now Showing movies
1.1.1.1: onCreateView()
1.1.1.1.1: fetchStoreItems()
1.1.1.1.2: request JSON array
1.1.1.1.3: list of Movie items
1.1.1.1.4: notifyDataSet Changed()
2: click one of movie titles/posters
2.1: onClickListener()
2.1.1: startActivity(Movie DetailsActivity)
2.1.2: directs to selected movie page
2.1.2.1: onCreate()
2.1.2.1.1: getIncoming Intent()
2.1.2.1.2: notify UI
3: click view show times
3.1: directs to movie showtimes page
3.1.1: onCreateView()
3.1.1.1: request JSON array
3.1.1.2: showtimes across different cinemas
3.1.1.3: notify UI

Dialog Map


select login
Login
log out
user's gmail or password are not correct
user's gmail and password are correct
View Account Profile
select view account profile
return to main page
Main Page
return to account profile
select viewsupport
cancel
GiveAppReview
select give app review
return to main page
ViewSupport
select view now showing info
return to main page
View Now Showing Info
direct to movie
cancel
select book movie ticket
book movie ticket
Book Movie Ticket
improper data
return to main page
return to main page
return to main page
return to main page
select view promotion
select upcoming info
select search movie
View Promotion
View Upcoming Info
Search Movie

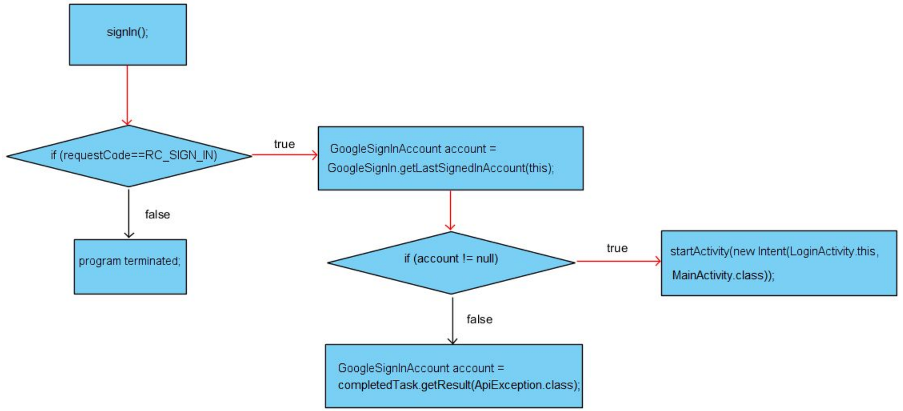
Application Testing

Black Box Testing

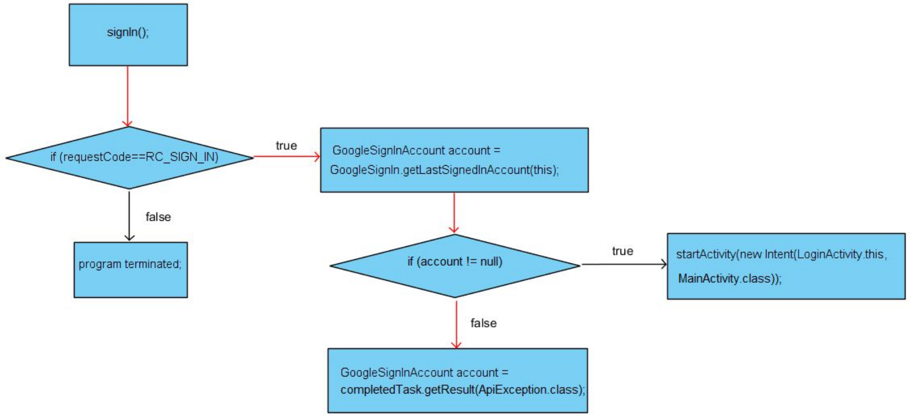
Test Priority (Low/Medium/High): High
Test Title: Movie Details
Description: To test movie details of a specific movie

Steps	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1	User clicks on Movie logo on bottom navigation		A list of now showing movies will be shown. The list contains an image of the movie and the movie title.	A list of now showing movies will be shown. The list contains an image of the movie and the movie title.	Pass
2	User clicks on a movie from the list	*"Maleficent"*	Movie details of *"Maleficent"* will be shown: rating,language, synopsis, director, cast, duration and NA if not available. There will be a "VIEW SHOW TIMES" button at the bottom of the page for user to click.	Movie details of *"Maleficent"* will be shown: rating,language, synopsis, director, cast, duration and NA if not available. There will be a "VIEW SHOW TIMES" button at the bottom of the page for user to click.	Pass

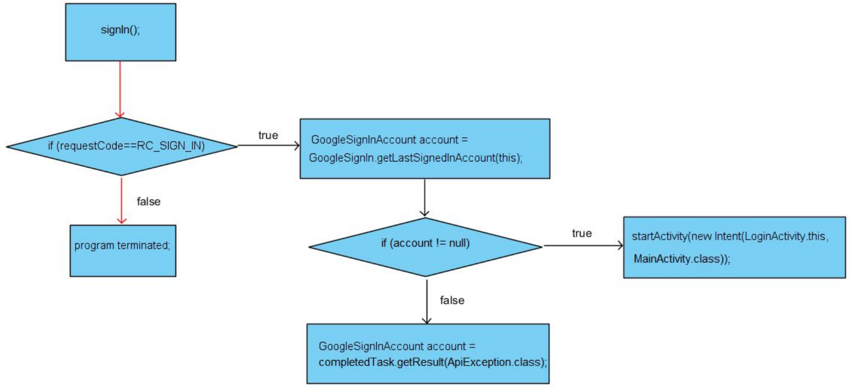
White Box Testing: *Login*



signIn();
if (requestCode==RC_SIGN_IN)
true
false
program terminated;
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);
if (account != null)
true
startActivity(new Intent(LoginActivity.this, MainActivity.class));
false
GoogleSignInAccount account = completedTask.getResult(ApiException.class);

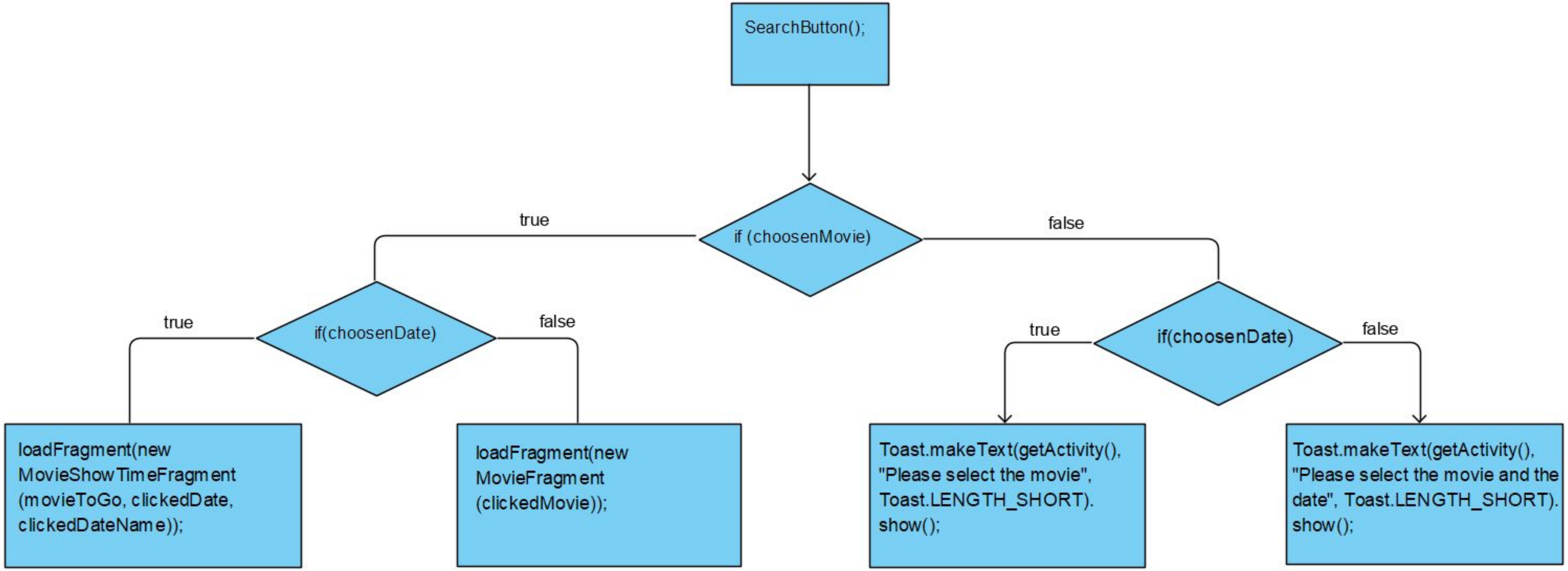


signIn();
if (requestCode==RC_SIGN_IN)
true
false
program terminated;
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);
if (account != null)
true
startActivity(new Intent(LoginActivity.this, MainActivity.class));
false
GoogleSignInAccount account = completedTask.getResult(ApiException.class);



signIn();
if (requestCode==RC_SIGN_IN)
true
false
program terminated;
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);
if (account != null)
true
startActivity(new Intent(LoginActivity.this, MainActivity.class));
false
GoogleSignInAccount account = completedTask.getResult(ApiException.class);

White Box Testing: *SearchMovie*


SearchButton();
if (choosenMovie)
true
false
if(choosenDate)
true
false
if(choosenDate)
true
false
loadFragment(new MovieShowTimeFragment (movieToGo, clickedDate, clickedDateName));
loadFragment(new MovieFragment (clickedMovie));
Toast.makeText(getActivity(), "Please select the movie", Toast.LENGTH_SHORT). show();
Toast.makeText(getActivity(), "Please select the movie and the date", Toast.LENGTH_SHORT). show();

Application Live Demo

Learning Points

- Get hands on experience on SDLC: Requirement elicitation, requirements analysis, conceptual & dynamic models, implementation and testing
- Learn to apply software engineering practices to develop Android application instead of normal web application
- Get the chance to develop an application that is useful in daily life, especially in Singapore

Challenges

- Little to no experience in developing Android application
- Sometimes found it difficult to integrate codes since everyone is involved in code implementation
- Need to continuously update/refine design models

Thank you! :)