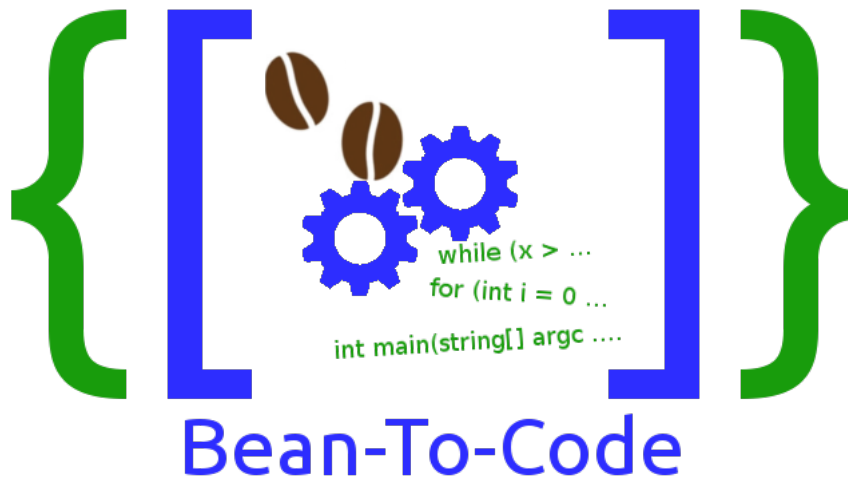# COS 301 Software Documentation

Melany Barnes 12030466
Dieter Doman 11002566
Johan Esterhuyse 10043285
Rudiger Roach 11004322

Version 1.0
GitHub link: https://github.com/RudigerRoach/301_main_emma.git

Contents

1 Vision and Scope

1.1 Vision

Our client creates software for camera club event management. A big part of an event comprises of an image judging process. Currently the process is completed by using Infra-red remotes and receivers, but this configuration is limited in terms of usability and the amount of judges that can judge concurrently.

The proposed solution will replace the hardware remote with a software application to run on a mobile device. The mobile application should alleviate all of the issues caused by the current setup, and should be developed with a server component that plugs into the existing EMMA system.

1.2 Scope

Create a software solution that:

- Runs on IOS and Android mobile devices.

- Allows as many as 20+ judges on the night.

- Allows judges to register against the event (in order to score) by capturing an email address.

- Remembers the scoring device for future meetings such that registration is not required again.

- Caters for realtime scoring.

- Can display a thumbnail image of that currently being judged.

- Caters for simple score entry bound within a variable range, as well as pluggable scoring methods that could include boolean scoring.

- Reports meaningful error messages, in a clear way.

- Allows for quick correction and re-capture.

- Can notify a judge of outstanding scores.
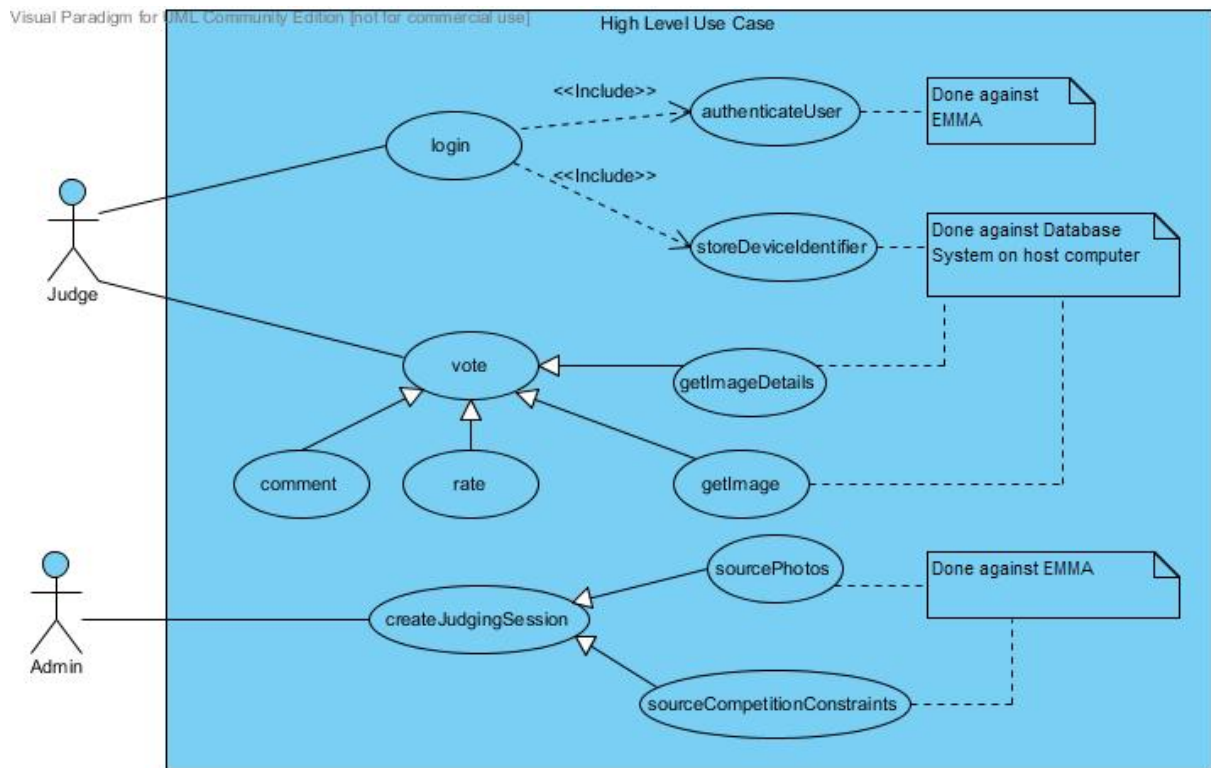
High Level Use Case Diagram

2 Architecture requirements

2.1 Architecture requirements

2.1.1 Architectural scope

- Provide an infrastructure for a judge to rate photos on a mobile device.

- Provide a database to link a judge's phone id to his email address.

2.1.2 Quality requirements

- Usability
  99 % of users should be able to use the system with little to no prior training.

- Scalability
  The deployed system must be able to operate effectively under the load of 50 concurrent users. The 50 users will be handled by Jetty which creates a thread for each person that connects to certain servlet at a point in time thus it will be able to handle the concurrency. It will be tested by creating a thread pool of 50 threads and doing 50 unit tests concurrently.

- Installability
  It should be easy to install the server side component and the effort to get it running each club night should be minimal. There will be a computer at the event running the EMMA server component and our server should be installed on it. The application should also be easy to download and install on the judges phone.

- Performance requirements
  All operations on application should respond within less than 1 second.

- Testability
  All services offered must be accompanied by unit tests. The tests should ensure that all pre-conditions are met before the service is delivered and that all post-conditions are met after the service has been delivered.

- Security
  The systems functionality should be only available to users who can be authenticated through the EMMA system. The users email address will only be used with no password. New users have to create an account before being granted access to the application if the sessions is closed. If the session is opened any person should be allowed to use it.

2.1.3 Integration and access channel requirements

- Integration requirement
  The production version of this application will need to integrate with EMMA. EMMA is Java a based application.

- Access channels
  The mobile application will have to go through a web-service which will be the public interface for the server-side component. It uses HTTP communication between then mobile and the server Jetty component.

2.1.4 Architectural constraints

- The mobile application should run on Android and iOS operating systems.

- The PC's that will be running the server side of the application and EMMA component will generally not be the latest technology(limited memory and processing power).

- There will be limited to no internet connection.

- The communication between the mobile device and server PC will be done over a wifi network.

- The server side component of this project should be able to run on Windows and OS X operating systems.

2.2 Use of reference architectures and frameworks

- JIRA Framework for the SCRUM agile method.

- Appcelerator Titanium framework which is an open-source software development kit for cross-platform mobile development.

- Jetty for hosting server and servlet handling.

- TiJasmine framework to run javascript unit testing in the Titanium framework.

- JUnit for java unit testing framework.

2.3 Technologies and languages

- Java

- JavaScript

- XML

- Microsoft Access Database

3 Functional requirements and application design

3.1 Introduction

This section discusses the functional requirements for the mobile judging system.

3.2 Required Functionality

3.2.1 Login and Auto Login

To login for the first time a user will have to enter his email address. The email provided will be authenticated by EMMA. If login fails the user will be informed that he is not registered to be a judge for the current session. If login is successful the device's unique identifier will be sent to the server to be stored in the database so that the device can be remembered on the system. This will allow for auto login - if a user attends a session where he is able to judge his phone will automatically be logged into the system when he enters the application. The user will then be able to use the rest of the system.
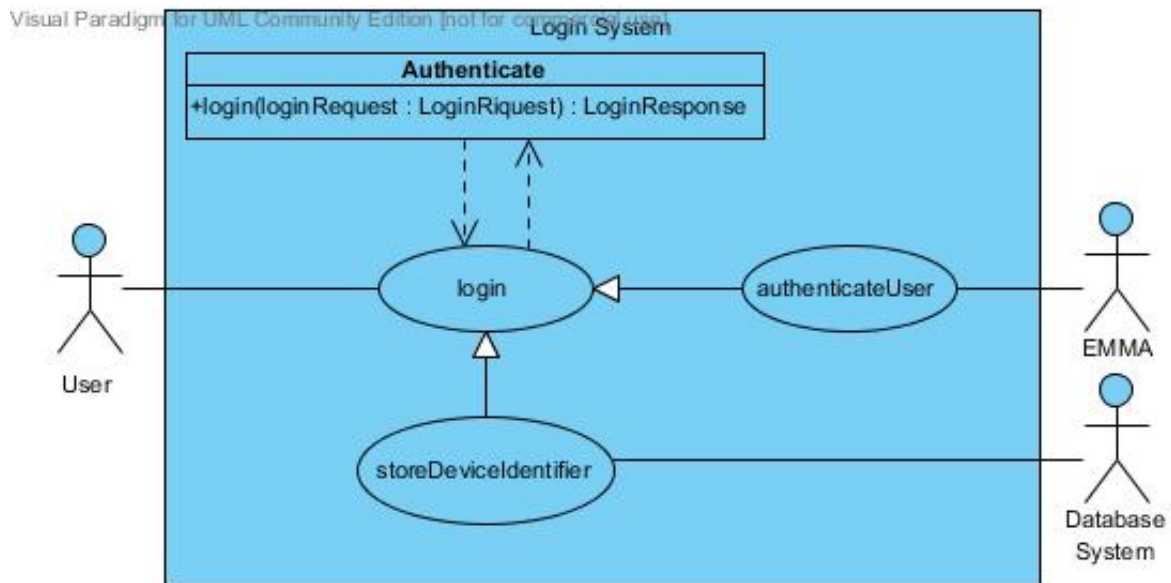
Login Use Case Diagram

3.2.2 Create Judging Session

When the server is started, it will request that the session's photos as well as all the competition constraints be sent to it. The constraints will contain the type of session (Open event, Closed event, Yes/No, Winner), the range for a valid score and if comments are enabled.

CreateJudgingSession

**JudgingSessionGenerator**

+createJudgingSession(competition : Competition) : createJudgingSessionResponse

createJudgingSession

sourcePhotos

Competition
Host

EMMA

sourceCompetitionConstraints
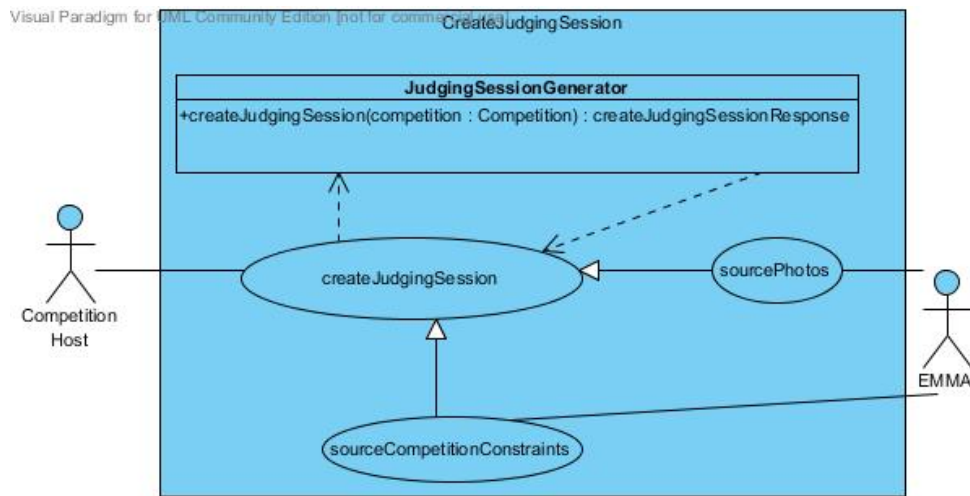
Create Judging Session Use Case Diagram

3.2.3   Voting

If a user logs in before the event starts, a loading screen will be displayed until
the event starts. The server will inform the user's application when the event
starts. The server will then pass through the first image and the details about
the image. The details will contain the image name, the bottom and top score
ranges as well as if comments must be enabled. The users will vote for the image
and will be able to leave a comment if the comments are enabled. The server
will either have a time limit per image or the server will check if all users have
submitted their vote. If not all users have submitted their vote, the server will
notify those users. If all users have submitted their vote, the next image and its
details will be displayed. This will continue until all images have been scored.
The user will be notified that voting is done.

Voting System

rate

comment

vote

User

getImage

getImageDetails
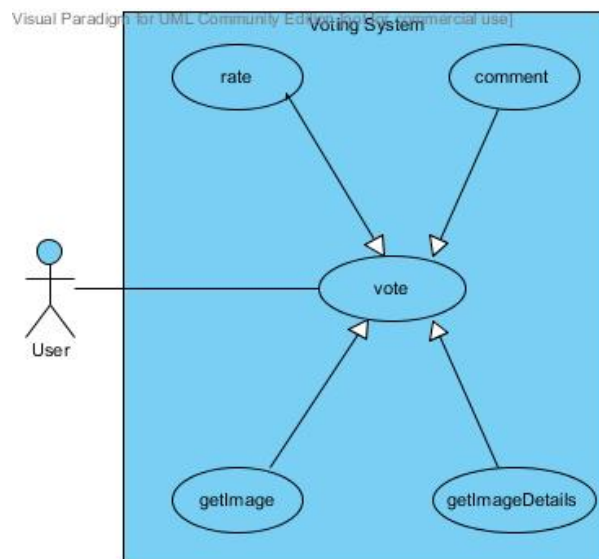
Voting Use Case Diagram

8

3.3 Use case prioritization

Critical Use Cases are the main cases that the system is made up of namely: Login, Create Judging Session and Voting. Without these cases the system will have limited to no functionality which will lead to a system that is not required by anyone.

Important Use Cases are the cases that improves the critical use cases and introduces a wider variety of functionality. These cases are Auto Login.

3.4 Use case/Services contracts

Services contract for Login

JudgingSessionGenerator

+createJudgingSession(competition : Competition) : createJudgingSessionResponse

{pre:Competition exists}
{post:JudgingSession generated}

GenerateJudgingSessionRequest

-competitionName : String
-sessionType : String
-commentsEnabled : Boolean

GenerateJudgingSessionResponse

Checks in the database if
a competition with this
name exists and if it does
generates a session for
this competition. If not an
exception is thrown

JudgingSession

-imageID : Long[]
-images : BufferedImages[]
-imageDetails : String[]
-judges : String[]
-scoringBottumRange : int
-scoringTopRange : int
-commentsEnabled : Boolean
-comments : Comment[]
-openSession : Boolean
-controlledSession : Boolean

+JudgingSession()
+getImageID()
+setImageID()
+setJudges()
+getJudges()
+getSessionType()
+setControll()
+getControll()
+SetRanges()
+getTopRange()
+getBotRange()
+setAllImageDetails()
+getAllImageDetails()
+setImageDetails()
+getImageDetails()
+getImages()
+loadImages()

Services contract for Create Judging Session

Judge

+vote(voteRequest : VoteRequest) : VoteResponse

{pre: valid JudgingSession assigned passed through}
{post: ratings within range of judgingSession}
{post: all images judged}
{post: final result obtained}

VoteRequest

-judgingSession

VoteResponse

-ratings : int[]
-comments : String[]

\<\<use\>\>

\<\<has a\>\>

VotingInterface
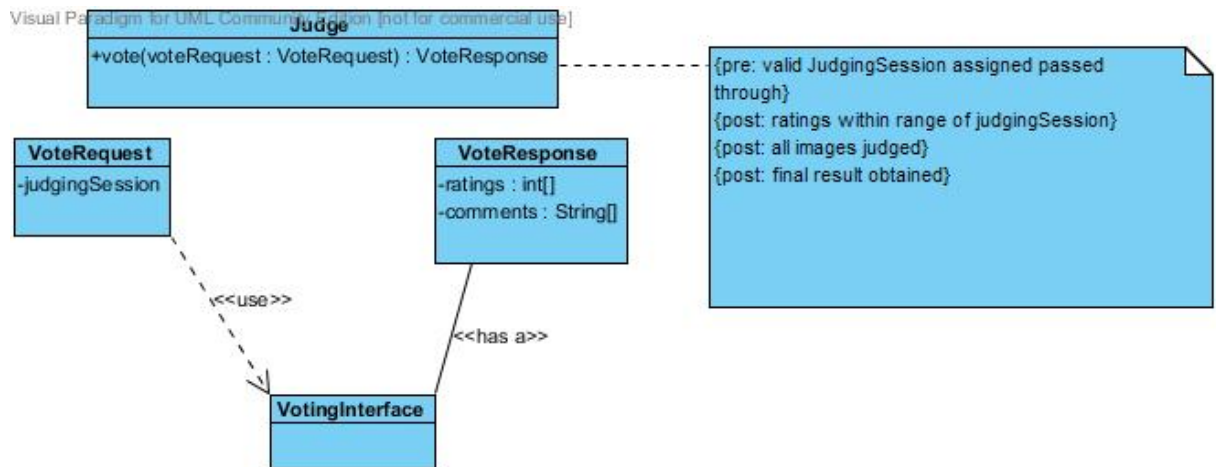
Services contract for Voting

3.5   Process specifications

loginProcessSpecification(loginRequest : LoginRequest) : LoginResponse

\<\<raiseException\>\>

couldNotAuthenticateUserException

authenticateUser
(EMMA::)

authenticatUserRequest

LoginRequest

storeDeviceIdentifierRequest

storeDeviceIdentifier
(DatabaseSystem::)

storeDeviceIdentifierResponse

loginResponce
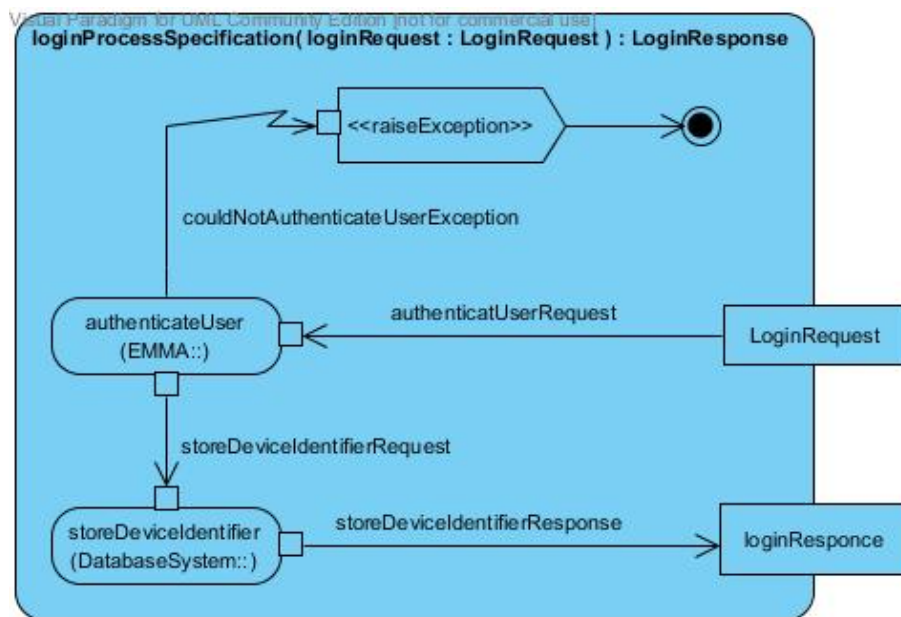
Login Activity Diagram
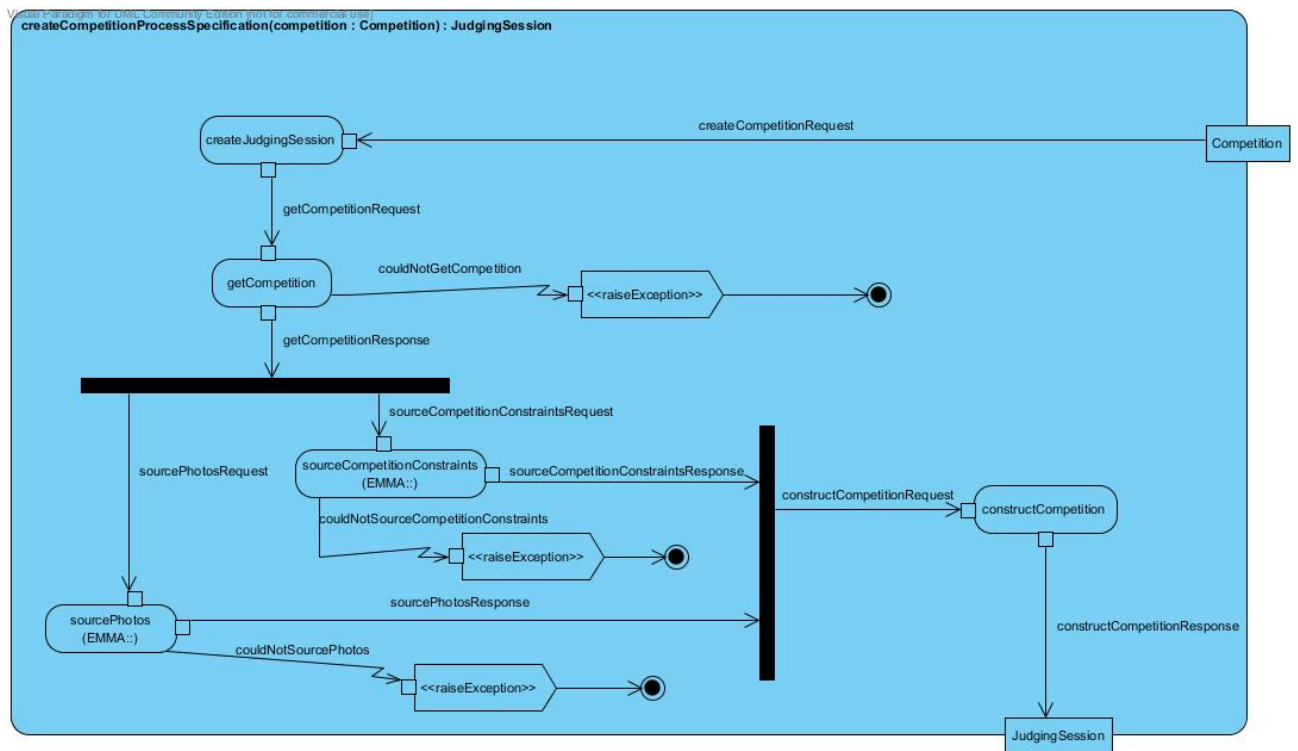
Create Judging Session Activity Diagram

VotingProcessSpecification(votingRequest : VotingRequest) : VotingResponse

retryRequest

[No]

Wait

votingRequest

sessionStartedRequest   sessionStarted   [Yes]   startVoting

proceedRequest

constructVote   AllVotersDone

constructVoteRequest   [No]   [Yes]

RateResponse   CommentResponse   retryRequest

Wait   VotingRequest

Rate   Comment

RateRequest   CommentRequest   addVote   Voting Complete   [Yes]   votingResponse

[No]

constructVotingPage   getImageDetailsRequest   getImage

getImageDetails

Voting Activity Diagram

3.6   Domain objects

4   Project management

4.1   Software development process

We followed the scrum software development methodology. Its agility, in terms of requirements, meant that the project was able to grow during development, while we where discovering new requirements from the client.

4.2   Issue management

On the clients suggestion, we used the Jira issue tracking system, combined with git and gitHub for version control.

4.3   Team profile

Melany Barnes:

- Team Lead.

- Developing the bulk of the mobile app front-end.

Dieter Doman:

- Developing the back-end of the server app.

Johan Esterhuyse:

- Developing the front end for the server app.

- Paying special attention to documentation and diagrams.

Rudiger Roach:

- Developing the back-end of the mobile app.

- Developing the database for the server app.

- Assisted in development of the mobile app front-end.

4.4   Project progress

We aimed to provide burndown charts for the entire project progress but unfortunately The issue tracking system ran from a server at the clients' residence and it experienced a lot of down-time. Luckily we can provide burndown charts for the following sprints:
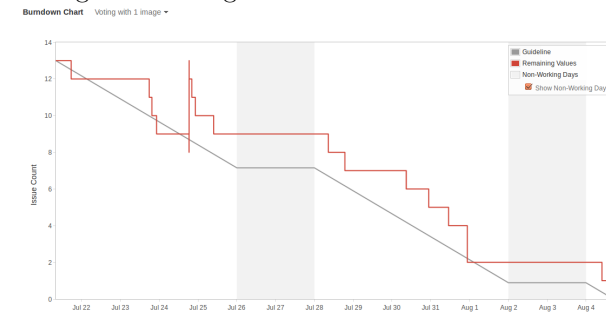
- Authentication
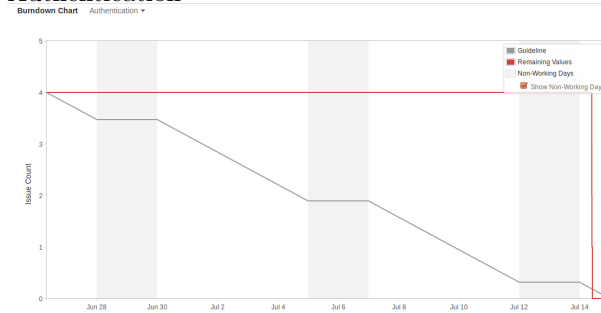
- Auto login

.

- Voting with 1 image
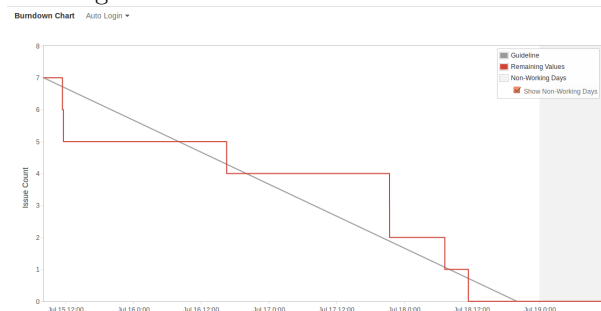
Burndown Chart — Voting with 1 image

4.5   un-implemented functionality

- Displaying results after voting has completed after.

- the mobile app is unaware of a server crash and will wait indefinitely in this case.

4.6   Main risks and challenges faced

- Getting to know Jira and issue tracking.

- Getting to know the Appcelerator platform.

- Getting to know JavaFX using FXML.

- Multi platform support.

- Multi resolution support.

5   Glossary

EMMA - Entry and Member Management Application
His - Refers to his/her
He - Refers to he/she