

CMPE 451.01 - Milestone 1 Report

Looking for Concerts

Group 2

November 3, 2017

Contents

1	Executive Summary	1
1.1	Description	1
1.2	Technical overview	2
1.2.1	Back-end	2
1.2.2	Front-end	2
1.2.3	Android (mobile)	2
2	Deliverables Status List	2
3	Evaluation of Deliverables & Their Impact on Progress	3
4	Coding work	4
5	Requirements Satisfied	5
5.1	Functional Requirements	5
5.2	Non-Functional Requirements	5
6	Design	6
6.1	Android	6
6.2	Web	6
7	Project Plan	6
7.1	Project Plan Table	6
7.2	Future Milestones	9
8	Code Structure and How We Use Git	9
9	Evaluation	9
9.1	Tools	9
9.2	Project Management	10

1 Executive Summary

1.1 Description

Looking for Concerts is a platform for musical concert-goers, where people from all ages come together to experience the warmth of a live concert stage with thousands of other music lovers. After a user signs

up, he can create concerts, follow or be followed by friends, search for concerts, see his friends' activity and attend concerts! Our intelligent recommendation engine will quickly pick up his interests. Especially if he connects his LFC account with Spotify. What's more, he can find past concerts and get lost in the comments, upvoting, downvoting and reacting! We would like to provide an immersive experience and help concert-goers find both exciting concerts to go to and interesting people to go to those concerts with.

1.2 Technical overview

1.2.1 Back-end

Our Looking for Concerts (LFC) API is powered by the Django framework. Kemal Berk and Haluk were in charge of creating the class models based on the class diagram on our Wiki page and implementing the endpoints. There were minor modifications in classes. For example, we do not have a class called "Concert Page" in our API, since it will actually be a part of the front-end. The endpoints implemented as of the current milestone are: signup, login, logout, delete user, get user, get all users, get user token, create concert, get all concerts, get/modify/delete a specific concert and comment on a specific concert. The passwords are hashed before being sent to our SQLite3 database and a unique user token is generated upon signup. Creating a concert or commenting on one requires a user to be logged in, implemented through token authentication. Deleting a user is forbidden at the moment. For the next milestone, we will be giving authentication only if a user wants to delete his own profile.

1.2.2 Front-end

For front end development, we used Angular 2 because our project will grow in size and workload over time and Angular provides an efficient and easy way to develop complicated applications as in our case, if applied correctly. We first struggled with Angular 2 as it is harder to learn than other frameworks (based on researches on forums), but finally managed to put together basic features such as login, sign-up, create and view concert in detail. We consider switching to React as our main front-end framework since it's easier to learn and apply, for the rest of the development process.

1.2.3 Android (mobile)

For Android, we used Java to develop our application. Elif, Furkan and Pinar were responsible for developing the app. Right now, with our application a registered user can login, create a concert, and comment on a concert. All users(registered or non-registered) can view all concerts in the system and see details of a concert. The design of our application is not good and we plan to improve it so that it is user friendly and easy to use. For now, we also require the user to know the exact location (venue and coordinates) of a concert while she is creating it, however for the next milestone, we will use Google Maps API and the user will be able to select the location she wants. We will give the user a chance to connect their account to Spotify. We will also add semantic tags. Right now, users just enter tags, but they don't mean anything. For the next milestone, we will search for those tags and the user will be able to select the tag she wants to add.

2 Deliverables Status List

1. Project requirements : ✓
2. Project plan: ✓
3. Class Diagram: ✓
4. Amazon AWS deployment: ✓
5. Features:

Feature Name	Description	Back-end	Front-end	Android
user signup	a new user enters credentials and is inserted in our system	✓	✓	✗
user login	an existing user logs in using his credentials	✓	✓	✓
user logout	a logged in user logs out to end his session	✓	✗	✗
concert creation	a logged in user creates a new concert	✓	✗	✓
view concert details	a user sees all the details of a specific concert, including comments	✓	✓	✓
comment on a concert	a logged in user comments on a concert	✓	✓	✓
view all concerts	a user sees all the list of concerts in homepage	✓	✓	✓

3 Evaluation of Deliverables & Their Impact on Progress

1. Project requirements: Almost finalized. If minor changes come up after customer meetings we might modify the requirements.
2. Project plan: Still needs revision. We might need task redistribution and stricter checks. We have 4 weeks until the next milestone. We can have a "checkpoint" each week and a "half-milestone" at the end of week 2.
3. Class diagram: This is very crucial since all the database depends on it. Right now the diagram in our Github wiki is not updated. We currently have a good structure though, and no crucial change will take place.
4. Amazon AWS deployment: Our API is deployed, but the front-end is not. We need to deploy it soon!
5. Features:
 - (a) User signup: This is a crucial feature since our application is a social platform. Users should be able to signup to benefit from the full content. Our Android version still does not have this functionality but will definitely have it very soon. We are thinking of switching to JSON Web Token (JWT) Authentication on the back-end before our second milestone.
 - (b) User login: Along with signup, login is also as important. Both our Web and mobile versions have the functionality.
 - (c) User logout: A user needs to be able to log out when he is done with using the application so that no one else will be able to perform actions on his behalf meanwhile. Logout is required for sure. We will be implementing it first before anything else!
 - (d) Concert creation: The most vital feature of our application. We have it on Android but not on the Web version. It is on the critical path of our plan and not delivering it hindered our progress. Will be implemented shortly.
 - (e) View concert details: Completed both on Android and Web. We will add more fields to Concert object, which will make the concert pages richer.
 - (f) Comment on a concert: Commenting is possible on both Android and Web. Comments are a huge part of user interaction.
 - (g) View all concerts: Available on both Android and Web.

4 Coding work

Group Member	Team	Contribution
Haluk Alper Karaevli	Back-end, Front-end	Learned Django, had the basic knowledge of Angular2. Implemented create Concert, get Concert, get All Concerts, Create Comment end points. Implemented Concert, tag, location, comment models. Constructed relations between Comment and Concert, Comment and User, Location and Concert, Tag and Concert. Implemented token generation. Looked into token authentication with Berk. Wrote API documentation of the back-end endpoints. In the front-end, created the service for passing token key between page components, corrected the connection between the pages using Angular2's Routing mechanism, solved minor visual bugs.
Kemal Berk Kocabağlı	Back-end	Learned Django and Sourcetree; Implemented the Registered User class and signup, login, logout, delete user, get user, and get all users endpoints; Built the user authentication system including authentication enforcement on concert and comment creation and password hashing; looked into token authentication with Haluk; commented the whole backend code; wrote the base LaTeX code for milestone report 1 and put the report together.
Pınar Baki	Android	Learned Android, Retrofit, Google Gson Library. Implemented login system for Android. Worked on token authentication in Android. Looked into concert list page design with Elif.
Elif Güler	Android	Learned Retrofit and Google Gson Library. Implemented the logic and design of the concert list page, where users can see all the concerts and the create concert page, where users can create a new concert. Implemented the get request method for getting all the concerts from our API and the post request method for creating a new concert.
Furkan Şenharputlu	Android	Learned Retrofit and Google Gson Library. Contributed to get request to get all concerts. Implemented concert details page, where users can see the details of a concert. Also, users can see all comments and they can make comment.
Halil Ozan Akgül	Front-end	Learned Angular 2 and used it to create Concert Detail Page of the project. Connected the database to front end to show the concert details. Created the HTML page and used CSS to design it. Also created some dummy concerts and comments to populate the database.
Alpertunga Ertin	Front-end	Timely deployment of backend code on AWS EC2. Concert creation page and login page implementation with Angular. Connection of different pages into one with Angular Router mechanism.
Oğuzhan Göller	Front-end	Did the task of implementing main page of our project. Worked generally angular and completed useful parts of tutorial on angular.io web page. Connected front end with concert api database via angular http format to get and list all concerts on webpage. Implemented general background look of web site and worked on css to provide a better appearance. For now, everything runs fine on main page.
Burak Sofu	Front-end	Learned Angular 2 basics via tutorial, implemented registration page. Gathered all front-end pages to rearrange file system according to Angular 2 standarts, and gave it its final form to be deployed.
Enes Hecan	Deployment	Created an Amazon EC2 instance and run. Updated server and setup tools like Django, virtual environment, python etc. to deploy our backend. Cloned our repository and run Django application on the server. Provided server information to frontend for a proper communication mechanism.

5 Requirements Satisfied

5.1 Functional Requirements

1.1.1.1 Registered users shall sign-in using their sign-up information.

1.1.1.2 Registered users shall stay signed-in until they logout.

1.1.1.3 Users shall be able to use their Facebook account to sign up.

1.1.5.1 In the concert page, users shall be able to see the artist, venue, concert date, tags, rating and comments.

1.1.5.2 Users shall be able add semantic tags about genres, artist's country of origin, etc. by entering text and then choosing from the presented options retrieved from a 3rd party semantic tag repository such as Wikidata. **Partially satisfied. Needs integration to a semantic tag repository**

1.1.5.4 Users shall be able to comment on a concert.

1.2.2.1 In the concert pages, non-registered users shall be able to see the artist, venue, concert date, tags, rating and comments.

1.2.2.3.3 Non-registered users shall not comment on concerts.

2.1.1 System shall save the user's login information, concert history, liked artists liked genres, followers and the users they follow.

2.1.2 System shall check whether each user e-mail is unique.

2.1.3 When a user is signing up, system shall require a name, an email adress and a password. Adding a profile picture, age and gender should be optional.

2.2.1 System shall require information about the artist, venue and concert date during the creation of a concert page.

5.2 Non-Functional Requirements

1.3 The language of the application shall be in English since it is the most widely used language in the world.

3.1 Database system shall be protected from unauthorized access.

3.3 User account email shall be unique.

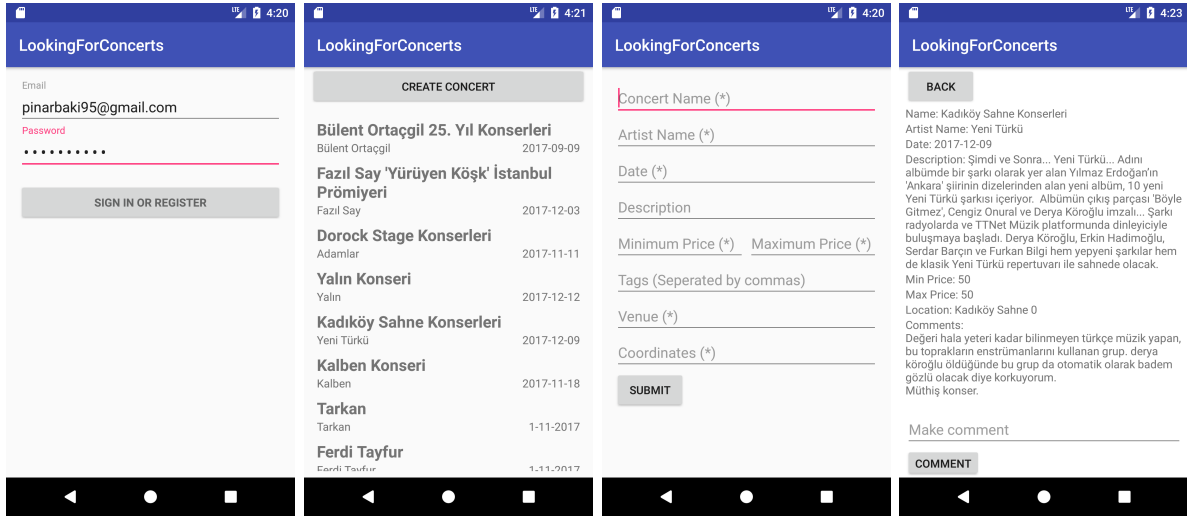
4.1.1 Android version shall be available to versions 4.4 (KitKat) and higher.

4.1.2 Web version shall run on latest versions of Google Chrome, Mozilla Firefox browsers.

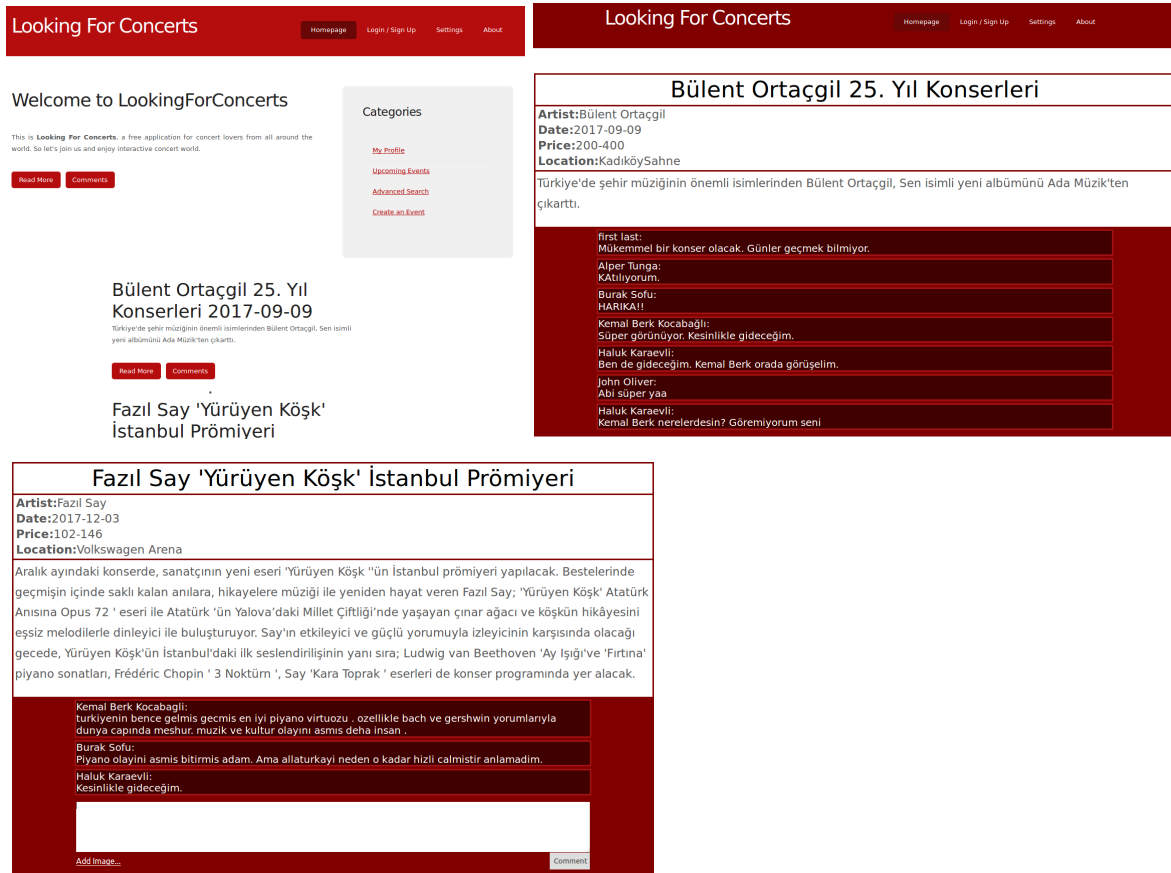
5.2 There shall be no noticeable lag, scrolling should be smooth with a good internet connection.

6 Design

6.1 Android



6.2 Web

































7 Project Plan

7.1 Project Plan Table

Below is the part of our project plan for this semester.

Looking4Concerts

ID		Task Mode	Task Name	Duration	Start	Finish
45			Big Revision	16 days	Fri 29.09.17	Fri 20.10.17
46			Plan Review	16 days	Fri 29.09.17	Fri 20.10.17
47			Task Distribution	16 days	Fri 29.09.17	Fri 20.10.17
48			Responding on Feedbacks	16 days	Fri 29.09.17	Fri 20.10.17
49			Database	0 days	Sat 21.10.17	Sat 21.10.17
50			Implementation	42 days	Mon 2.10.17	Tue 28.11.17
51			Back-End Developments	33 days	Mon 2.10.17	Wed 15.11.17
52			Database System Design / Implementation / Research	14 days	Mon 2.10.17	Thu 19.10.17
53			AWS Deployment	2 days	Tue 10.10.17	Wed 11.10.17
54			DB Modeling Search	3 days	Mon 2.10.17	Wed 4.10.17
55			User Table Implementation	7 days	Wed 11.10.17	Thu 19.10.17
56			Concert Table Implementation	7 days	Wed 11.10.17	Thu 19.10.17
57			API Implementation	6 days	Mon 2.10.17	Mon 9.10.17
58			Algorithm Implementation	17 days	Tue 24.10.17	Wed 15.11.17
63			Front-End Developments	37 days	Mon 2.10.17	Tue 21.11.17
64			Web Developments	37 days	Mon 2.10.17	Tue 21.11.17
65			Create Concert Page Implementation	9 days	Mon 2.10.17	Thu 12.10.17
66			Git Connections	2 days	Tue 10.10.17	Wed 11.10.17
67			Login Page Implementation	5 days	Fri 13.10.17	Thu 19.10.17
68			Search Page Implementation	5 days	Tue 24.10.17	Mon 30.10.17
69			Notifications Page Implementation	6 days	Tue 31.10.17	Tue 7.11.17
70			Search Results Page Implementation	5 days	Wed 8.11.17	Tue 14.11.17
71			User Page Implementation	5 days	Wed 15.11.17	Tue 21.11.17

Looking4Concerts

ID		Task Mode	Task Name	Duration	Start	Finish
72			Concert Page Implementation	9 days	Mon 16.10.17	Thu 26.10.17
73			Android Developments	42 days	Mon 2.10.17	Tue 28.11.17
74			Create Concert Page Implementation	9 days	Mon 2.10.17	Thu 12.10.17
75			Login Page Implementation	8 days	Fri 13.10.17	Tue 24.10.17
76			Concert Page Implementation	9 days	Mon 16.10.17	Thu 26.10.17
77			User Profile Implementation	5 days	Fri 10.11.17	Thu 16.11.17
78			Database Connection Implementation	8 days	Fri 17.11.17	Tue 28.11.17
79			Designing Android and Web UI	0 days	Fri 17.11.17	Fri 17.11.17
80			Documentation	9 days	Thu 16.11.17	Wed 29.11.17
81			Project Process Documentation	2 days	Thu 16.11.17	Fri 17.11.17
82			Tutorial Creation	5 days	Mon 20.11.17	Fri 24.11.17
83			Release	0 days	Wed 29.11.17	Wed 29.11.17

7.2 Future Milestones

1. Milestone 2

- (a) Logout, change password, edit profile information, delete account for users
- (b) Connect LFC account with Spotify
- (c) Registered user profile page
- (d) Attend/unattend concerts
- (e) Detailed concert ratings (location, costumes, music, stageshow) seen as bars
- (f) Add images to concerts and profiles
- (g) Follow/unfollow other users
- (h) Display friend activities on newsfeed
- (i) overall styling improvements
- (j) Integrate GoogleMaps API to location
- (k) Integrate a 3rd party semantic tag repository to tag selection when creating a concert

2. Milestone 3

- (a) Upvote/downvote comments
- (b) Report users
- (c) Report false concert information
- (d) Concert merge
- (e) Concert recommendation
- (f) Concert search
- (g) User search

8 Code Structure and How We Use Git

Different parts of the whole project stays in the repository separately. Each team works on their own folder, e.g. frontend developers work on lfc-frontend folder, backend developers on lfc-api.

We are developing every new feature in a new branch. When the time comes for this feature to integrate with the whole project, developer sends a pull request to the teammates related to this work. They evaluate, then either approves or denies by reasoning with a comment. A pull request can't be merged to the master branch if it is not approved by at least one team member.

9 Evaluation

9.1 Tools

For back-end, the tools we used are:

- **Django:** A framework of python to construct back-end of a project easily.
- **djangorestframework:** The main framework we use on top of the django in implementing back end. We used the serialization part of the rest framework to return object's corresponding json's in the back-end endpoints.
- **virtualenv:** A tool to create isolated Python environments. The basic problem being addressed by virtualenv is one of dependencies and versions, and indirectly permissions. For instance django were creating issues with the computers that installed panda.
- **iTerm2:** A Terminal replacement for MacOS. It is much more convenient to use.

- **Postman:** A GUI platform for API development. We used it to construct and cache our requests.

For front-end, the tools we used are:

- **Angular 2:** TypeScript we used for frontend.
- **HTML:** Markup Language we used to create the website.
- **CSS:** Style Sheet language we used for design.

For Android, the tools we used are:

- **Android Studio:** the IDE we used to develop our app.
- **Retrofit:** A type-safe HTTP client for Android and Java. We used it to write easier request methods.
- **Google Gson Library:** A Java serialization/deserialization library to convert Java Objects into JSON and back. We used it to serialize and deserialize our dto objects.

For project planning, the tool we used is:

- **MS Project:** A project management software, developed by Microsoft.

9.2 Project Management

- We could have managed the time more efficiently.
- Communication between teammates was sometimes poor. We need to be more clear and specific about the issues we are dealing with.

References

- [1] <https://angular.io/guide/quickstart>
- [2] <https://github.com/google/gson>
- [3] <http://square.github.io/retrofit/>
- [4] <https://www.djangoproject.com/>