**She codes;**

**Design document**

**Introduction**

**Scop**

This web application will be used for planning events such as birthday parties, bachelorette parties, or any other event that requires prior planning. The app is designed for the planners of the event and will provide a comfortable way for multiple planners to monitor all the necessary things for a successful event.

**Overview**

In this document we will start by explaining about the different uses and features of the application. We will describe in detail the apps design and how it integrates with other systems. Last, we will show how the app will be tested and ready for the users to start enjoying it!

**Terminology**

Planner – user of the app who has an event planed.  
Invites – people that are invited to the planners' event.  
Attending – Invites that confirmed arrival to the planners' event.

**Software design description**

**General flow**

When first opening the app, a list of all events that the user was/is planning is displayed. By pressing one of the events, the user will access a page which includes 5 options: invites list, attending list, refreshments list, equipment list and expenses list.

Invites list will include all the people that were invited to the event. Next to each name, there will be an option for the user to mark the person as coming, and his name will atomaticly be passed to the attending list. The refreshments list will be a table with two columns – one for the food, and the other for the person who is responsible to bring it. The equipment list will contain 3 tables, one for equipment needs to be bought, other for equipment that was bought, and another for equipment someone brings from home. Like the refreshments table, the second column in each table will include who is responsible for buying/bringing each item. The user could drag items from one table to another. The expenses list will allow the planners to write which planner paid for what, and then the total sum for each planner, and how much money the other planners owe him (or vice versa) will be displayed at the end of the page.

software architecture

Login

Sign in

**My app**

 Creating account for new users   
(maybe with Gmail or Facebook)

Authentication

For an existing user

Event list

|  |
| --- |
|  |
|  |
|  |

Invites list

|  |
| --- |
|  |
|  |
| תיבת סימון עם v עם מילוי מלא |

Attending list

|  |
| --- |
|  |
|  |
|  |

Refreshments

|  |  |
| --- | --- |
| brings | food |
|  |  |
|  |  |

 Equipment

Bought:

|  |  |
| --- | --- |
| responsible | item |
|  |  |
|  |  |

Brought from home:

|  |  |
| --- | --- |
| responsible | item |
|  |  |
|  |  |

needs to buy:

|  |  |
| --- | --- |
| responsible | item |
|  |  |
|  |  |

1

 Expenses

|  |  |  |
| --- | --- | --- |
| Who paid | cost | item |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| Total cost paid | planner |
|  |  |
|  |  |

Planner 1 owns x$ to planner 2

Main functions:

* Sign in / sign out / Login user
* Event list: add event, delete event…
* Add other user as planner to an event
* Invite list: add a guest, remove guest, mark guest as attending
* Attending list: add/remove guest
* Refreshments: add/remove item and the person responsible to it
* Equipment: add/remove item and the person responsible to it (function for each table)
* Expenses: add/remove item, its cost and the person that paid,   
  calculate total cost for each planner, calculate balances

User interface

* create an event
* invite other user to co-plan an event
* maintain invites list with add and remove buttons
* mark invites as coming (will copy it to the attending list)
* maintain refreshments list with add and remove buttons
* maintain equipment list with add and remove buttons,  
  option to drag an item from one table to another
* monitor expenses with add and remove buttons

Data handling  
We will use mongo DB as the database for the application. The stored data will be in BSON format.   
We will save data about the users in the system with their identifying information. We will also save the events, and for every event the users that plan this event, and all the needed information in the tables described at the software architecture section. The saved data will look something like this:  
{ "Event\_id": 1,

"Planners": [  
 { "Planner\_id": 123,  
 "Planner\_name": "Noy"  
 }, { "Planner\_id": 124,  
 "Planner\_name": "Guy"  
 } ] ………

}

Integration with other systems

The software will interact with other systems such as systems to register a user to the app (maybe with Google or Facebook accounts). Also, it will require direct access to the databases because most of the information in them is exposed to the user. For communication protocol we will use HTTP.

**Tests and Monitor**

Tests  
To check that our app works properly, we will start by running the following tests:

* Create an event and check that it appears in the database with the right details (name of the event, name of the planner...)
* Add a name to the invites list. Check if it exists in the database. Then, erase it and verify that after deletion it is no longer exist.
* Add a name to the invites list. Mark near that name that this person is coming, and check if a copy is created for attending list.
* Add a name to the attendings list. Check if it exits in the database. Then, erase it and verify that after deletion it is no longer exist.
* Add an item + name to the refreshments list. Check if it exits in the database. Then, erase it and verify that after deletion it is no longer exist.
* Add an item + name to each one of the tables in the equipment's list. Check if it exists in the database. Then, erase it and verify that after deletion it is no longer exist.
* Add an item + cost + name to the expenses list. Check if it exists in the database. Then, erase it and verify that after deletion it is no longer exist.
* Add a few rows of item + cost + name to the expenses list. Check if the calculation of total sum to each planner is correct.

Next, we will check that all the above are shown correctly in the app for the users (for example, when adding an item, it is displayed in the correct location for the user). We will also do a performance test – in order to see that the app can endure a lot of users at the same time. Finally, we will check that the user interfaces is as it was designed (colors, sizes, text font, etc.).

Logs

Thing that are going to be logged: User sign in the app, adds an event, joins an event of other user, add/delete an item to each one of the tables. This will help us to detect bugs or problems in our app and learn statistics about the use of the app (how many people use the app a day for example). Because this app allows multiple users to plan together an event, we will keep for every item that was added the name of the planner that added it, and the time it was added at.

Alerts  
alerting rules:

* User deleted data which should not be accessible to him from the database.
* The app is overloaded with users.
* The database is overloaded. There is not enough memory to save all data.

A/B Testing

We will create a few versions, each one with a little different UI – size and colors of the buttons, background colors, size and font of text, etc. We will ask users to rate the app visibility, and the UI with the higher score will be chosen. We could also use the data about users log in the app to understand which version attracts more users.