Meeting-Scheduler

System Requirements Specification (SRS)

Mobile Dev Team

[‘brohemz’, ‘The Girl on the Floor’]

Summer 2020

Created: May 25th, 2020

Meeting-Scheduler: System Requirements Specification

Outline adapted from https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

Table of Contents

I. Introduction

1.1 Purpose

This document is meant to overview the requirements for building an application that coordinates optimal meeting times between the individuals of a group.

1.2 Document Conventions

SRS	System Requirements Specifications
ERD	Entity Relationship Diagram
DB	Database
OS	Operating System
Language	Programming Language
Framework	Programming Framework
UI/UX	User Interface AND User Experience
PK	Primary Key
FK	Foreign Key

1.3 Intended Audience

This application is meant mostly for tech savvy prosumers. However, there may be a future update where this is pushed onto the app stores. In that case, the average Joe may use this in his personal life for friendly skype calls, or Joanne may utilize it for her professional duties.

1.4 Project Scope

The purpose of this project is to make it easier to form on-the-fly meetings between small-to-medium size groups of people. Often-times, people will have issues coordinating their schedules so that the meeting accommodates everyone. We would like to solve that problem with an application that accumulates each member's schedule and outputs the optimal meeting time.

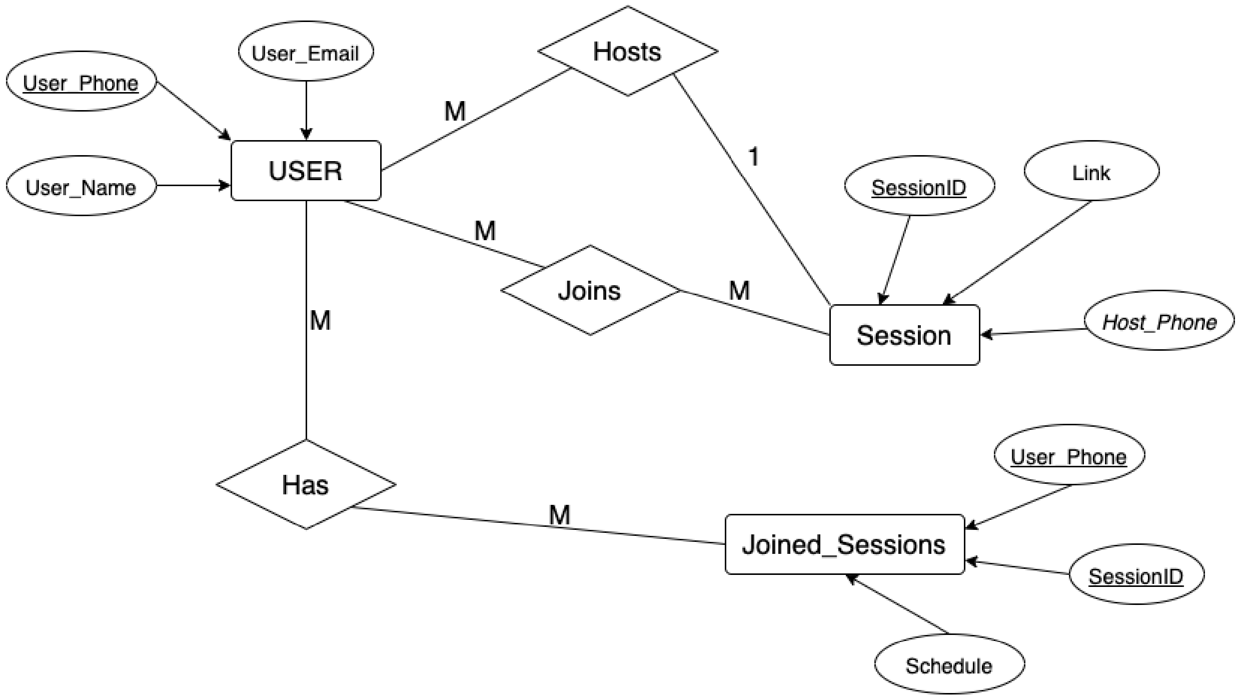
1.5 Minimum Viable Product

If this application makes it off the ground and serves its purpose of finding an optimal meeting time for a majority, then it will be a success.

II. Overall Description

2.1 Product Features

Entity Relationship Diagram



User:

- *User_Phone (PK)* - number | not null
- *User_Name* - varchar2
- *User_Email* - varchar2

Session:

- *SessionID (PK)* - number | not null
- *Link* - varchar2
- Description
- *Host_Phone (FK)* - number | not null

Joined_Sessions:

- *User_Phone (FK)* - number | not null
- *SessionID (FK)* - number | not null
- *Schedule* - varchar2

2.2 Use Cases

The users of this application will be split between one host and many members. The actions of each entity are detailed below:

Host:

- Host a session
- Manage active sessions
- View history of previously-created sessions
- Retrieve final meeting time and prepare final accommodations
- View all details of the session (member information)

Member:

- Join a session
- Input preferred meeting times
- View minor details of the session (amount of users, date created, meeting host)

III. Functional Requirements

3.1 Basic Functionality

Host will start a session by inviting group members to a shared "room" within the application. Each member will then input their preferred meeting times, and then app will return the optimal meeting time to the host.

3.2 Feature Overview

- Create a session
- Invitation to members (SMS or Link)
- Date / Time Picker Interface
- Database to store and accumulate this information

IV. External Interface Requirements

4.1 Backend Requirements

- Dart (Language)
- Firebase (DB)

4.2 Frontend Requirements

- Flutter (Framework)

4.3 Software Interfaces

- iOS

- Android OS

4.4 Hardware Interfaces

- iPhone OR Android Phone
- A minimum of 3G-speed network connection

V. Nonfunctional Requirements

5.1 Safety Requirements

In the event of a database corruption/destruction, redundant backups should be available to restore the database to a recently-working state. These backups should be created periodically and should be off-site if possible.

If the server/database is offline, it should appear offline. At no time should the software function without a live server online.

5.2 Security Requirements

User accounts should have a strong *Password Policy* that adheres to a strict *Password Standard.* This will ensure the safety of user accounts from dictionary attacks.

Validity tests will be run on all fields in order to prevent any malicious injections. These tests will check whether the information entered into the field matches an expected format.

All passwords and information will be encrypted and will not be visible in plain text to any admin of the system.

5.3 Software Quality Attributes

- All views, buttons, etc. should adhere to common UI/UX conventions
- Software shall run without hiccups, unless stall indicated by a loading symbol
- Functionality described in this document must be accessible to those without technical prowess

VI. Project Schedule

6.1 Sprint Timeline

#1: 06/01/2020 - 06/08/2020 | Research (i.e. Learn Flutter) / Design

#2: 06/08/2020 - 06/22/2020 | Server Cards, Some Backend Cards

#3: 06/22/2020 - 07/06/2020 | Finish Backend Cards, Start Fleshing Out Frontend

#4: 07/06/2020 - 07/20/2020 | Finish Frontend, Bring It All Together

#5: 07/20/2020 - 08/03/2020 | Finish It!!!

6.2 Product Backlog

- Learn the fundamentals of flutter
- Dive into advanced topics of flutter
- Setup Database (Backend)
- Both - Data Validation (Backend)
- Login (Backend)
- Host - Create Session (Backend)
- Host - Manage Session (Frontend)
- Host - Retrieve Optimal Time (Backend)
- User - Join Session (Backend)
- User - Input Schedule (Frontend)
- Both - Manage Session History (Frontend) (Backend)
- Both - Fix Bugs (Frontend) (Backend)
- Both - Final Touches (Frontend) (Backend)

6.3 Sprint Backlog

#1 Research / Design
- Design Document
- Learn the fundamentals of flutter
- Dive into advanced topics of flutter

#2 Server Cards, Some Backend Cards
- Setup Database (Backend)
- Both - Data Validation (Backend)
- Login (Backend)

#3 Finish Backend Cards, Start Fleshing Out Frontend
- Host - Create Session (Backend)
- Host - Manage Session (Frontend)
- User - Join Session (Backend)
- Both - Manage Session History (Frontend) (Backend)

#4 Finish Frontend, Bring It All Together
- User - Input Schedule (Frontend)
- Host - Retrieve Optimal Time (Backend)

#5 Finish It!!!
- Both - Fix Bugs (Frontend) (Backend)

- Both - Final Touches (Frontend) (Backend)