Version 1.0

09.11.2019

On-Site Coder (Developer Manual)

Presented by:

## Halİl etka tutkun

## Saİd furkan Ayvaz

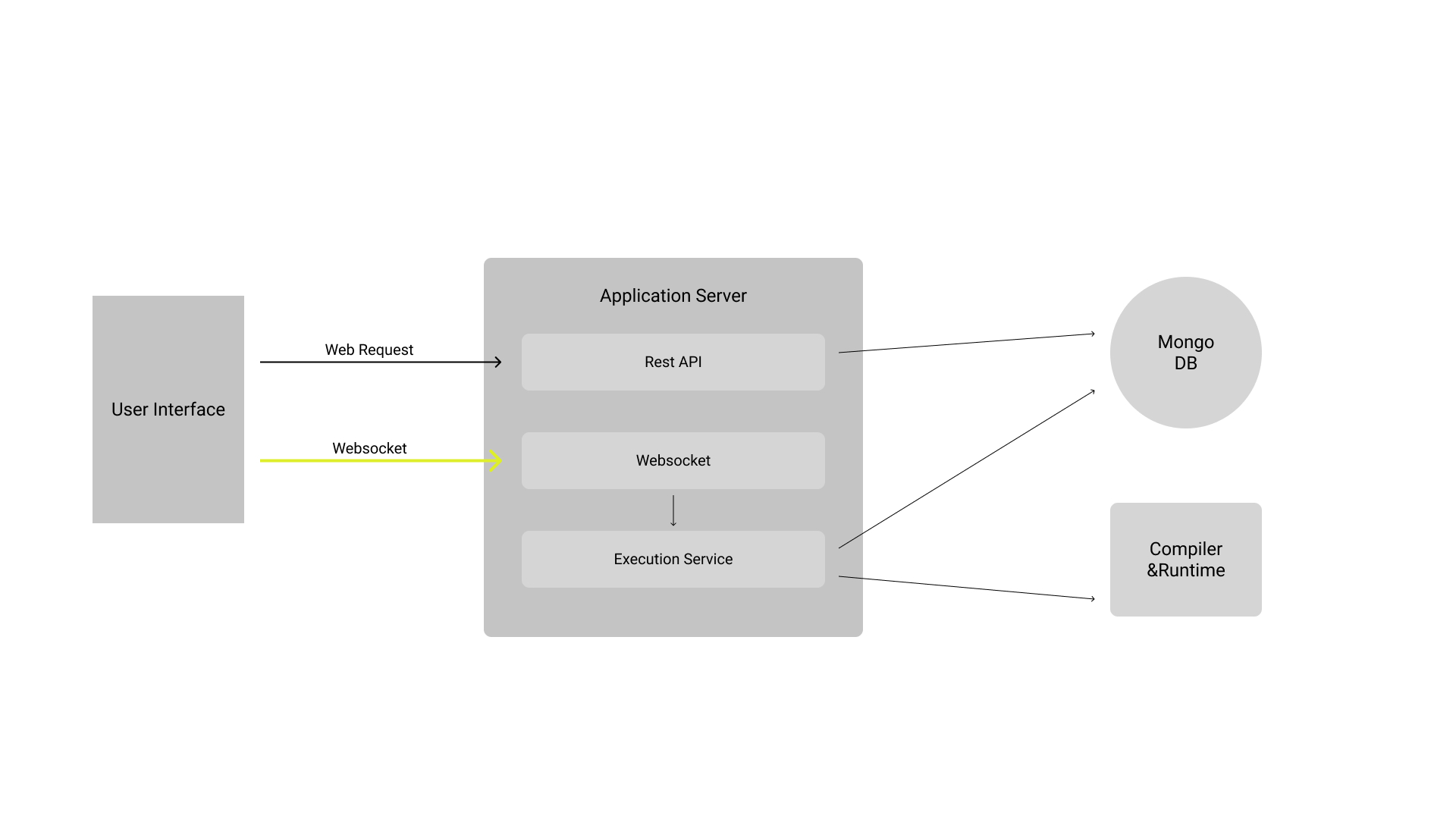
## Uğur küçükterzİ

# workflow overview

Below is the overview of the core functionality of the program. There might be other functionalities as well, but they will be explained in the upcoming sections.

* 1. Educator creates an account.
  2. Educator creates a challenge (public or private).
  3. Educator shares the challenge key with students
  4. Student creates an account.
  5. Student joins to a challenge through a shared key or by discovering public challenges.
  6. Student tries to solve the challenge and tests his/her solution.
  7. Educator checks the results of each executed student solution test.
  8. Educator confirms the final grading.

# System Archıtecture



# Functional requirements

## System user

### Account Management

* + - 1. Can choose to create an account as an educator or as a learner.
      2. Should validate email address.
      3. Can reset password with validated email.
      4. Can change email address.
      5. Can remove account.

## educator

### Challenge Creation

* + - 1. Can choose to create a public or private challenge.
      2. Should choose several programming languages the challenge will be solved with.
      3. Can restrict access to private challenges by an email domain that belongs to an institution.
      4. Can allow testing by uploading zip files for private challenges.
      5. Should create at least one test case with execution command and expected output.
      6. Test cases can be visible or hidden for students.
      7. Can create required input files.
      8. Can create a starting template.

### Solution & Result Review

* + - 1. Can review the output of the tests.
      2. Can review solution code.
      3. If the code is uploaded as zipped file should download it to review it.
      4. Can change the test score based on the reviews.

## learner

### Challenge Discovery

* + - 1. Can search by name.
      2. Can apply programming language filter.
      3. Can sort by creation date.
      4. Can sort by popularity.

### Joining to A Challenge

* + - 1. Can join to a challenge via the shared key.
      2. If there is an access restriction of a private challenge and learner’s mail address doesn’t match with the required domain name, he/she should get an error.
      3. Can discover public challenges to join.

### Solving the Challenge

* + - 1. Can solve the challenge in the environment provided by the system.
      2. Should be able to save the solution to the system.
      3. Can test the solution against the test cases created by the educator.
      4. Latest test results should be saved automatically as the score of the solution.
      5. Can solve the challenge in his own environment.
      6. Should upload the code in a zip file to test and submit.
      7. If file size is greater than 5mb should throw an error.

## System admın

### Checking Suspicious Behaviors

* + - 1. Should see the tests run more than the time limit.
      2. Should see the tests that gave a system error.
      3. Should see the tests that consume too much memory.

### Managing Accounts

* + - 1. Can blacklist accounts which has suspicious behaviors.

### Managing Challenges & Solutions

* + - 1. Can remove problematic challenges.
      2. All related solutions and files to the challenge should be deleted.
      3. Can remove problematic solutions.

# dataBase entıtıes

Below are the schemes of business entities that will be used in the system to store and query the required data. Scheme includes the property names and their types of each entity.

### User

* + - 1. Id – string
      2. Email – string
      3. UserName – string
      4. PasswordHash – string
      5. Role – enum
      6. IsBlocked – bool

### Challenge

* + - 1. Id - int
      2. OwnerId – int
      3. OwnerUserName - string
      4. IsPrivate – bool
      5. PrivacyDomain – string
      6. AllowSolutionUpload - bool
      7. Name - string
      8. Language – enum
      9. ProblemDefinition - string
      10. Files – [ { Id: int, ReadOnly: bool, Name: string, Data: string } ]
      11. TestCases – [ { Id: int, Name: string, Hidden: bool, Parameters: string, ExpectedResult: string } ]

### Solution

* + - 1. Id – int
      2. ChallengeId – int
      3. OwnerId – int
      4. OwnerUserName - string
      5. IsUploadedSolution - bool
      6. Language - enum
      7. Files – [ { Id: int, Name: string, Data: string } ]
      8. TestResluts – [ { TestCaseId: int, TestDate: date, StdOut: string, Score: int } ]

# Interface desıgn



