CS 492: Senior Design Project

Design Report

BilMate : Group T2335

Ahmet Salman	21901004
Atasagun Samed Şanap	21902435
Ebrar Bozkurt	21802824
Javid Moradi	21903645
Onuralp Avcı	21902364

13/03/2023

Jury Members: Erhan Dolak, Tağmaç Topal, Uğur Doğrusöz

Innovation Expert: Ahmet Kocamaz

1. Introduction

1.1 Purpose of The System

One of the major struggles for university students, particularly in Turkey, is finding acceptable housing. Although there are student housing options, they are sometimes overcrowded, in poor shape, pricey, or located far from the university. Students now have to find suitable housing and roommates on their own as a result of this. While some online tools are available to assist with this, such as "Bilkent Itiraf," they are not designed specifically for the task. They may not be successful in pairing people based on their criteria.

BilMate addresses this issue. Students looking for roommates and those looking for shared housing can connect on this app's social network and search through choices depending on their interests. Users of the app will be able to enter information about their chosen location, number of roommates, rent, smoking policies, pet allowance, and other preferences. The app will match up house-seekers and housemate-seekers based on this information. Users' chances of discovering a compatible match increase with the number of data fields they fill out.

1.2 Design Goal

1.2.1 Usability

Offering its users a simple and user-friendly interface is one of BilMate's main design objectives. The intricate algorithms used by the app will be concealed behind the UI layer to prevent users from becoming confused by the intricate details. To give customers a seamless experience, the questioning flow—which is essential to the matching algorithm's accuracy—will be enhanced. The user interface of the app will be made to be simple to use and comprehend so that it may be used by a variety of users.

1.2.2 Security

The protection of the sensitive data belonging to BilMate's users is another important design objective. Using a "data protection by design and default" development approach, the app will be created with both backend and frontend security in mind. Data that does not need to be stored will not be stored at all, and any sensitive data will be secured with a strong encryption technique.

1.2.3 Scalability

Scalability is a crucial design objective for BilMate. The software must be scalable enough to handle the growing user traffic as the number of Bilkent students and users from other colleges

increases. The architecture of the app will be built to support a huge user base without compromising on performance or user experience.

1.2.4 Maintainability

Maintainability is BilMate's ultimate design objective. The app's user traffic will fluctuate throughout the year, peaking at the start of each semester. The architecture of the app must enable developers to quickly address any problems that may arise and maintain the app's functionality and user experience. As the backend and front end will be integrated, upgrades and corrections may be made quickly and effectively.

1.3 Definitions, Acronyms, and Abbreviations

- BilMate: BilMate is a mobile application that is intended to assist users in locating suitable housemates and lodging options according to their tastes and way of life.
- House Sharer: A person who splits rent and utilities in a home or apartment with one or more other people.
- House Seeker: An individual who is actively looking for a home or apartment to rent or cohabitate in.
- App: The term "application" is shortened to just "app," and it often refers to a software application created to run on a mobile device like a smartphone or tablet.
- API: An acronym for "Application Programming Interface," or API, denotes a collection of tools and protocols used to create software programs.
- UI: The term "User Interface" or "UI" is an acronym for a software application's graphical and interactive elements that enable user interaction.
- UX: An acronym for "User Experience" which refers to the total interaction between a user and a software product, including factors like satisfaction, usability, and ease of use.
- ToS: An acronym for "Terms of Service" which refers to the set of rules and restrictions placed on the users of BilMate for the purposes of maintaining order and safety of all the users.

1.4 Overview

BilMate is a mobile app created to help university students, particularly those in Turkey, discover acceptable lodging and roommates. The app connects users with people looking for shared accommodation or roommates based on preferences, including location, number of roommates, rent, smoking restrictions, pet allowance, and other factors. By offering a social network and a matching algorithm that improves the chances of finding a compatible match, BilMate seeks to alter the way college students choose their roommates and find suitable lodging.

BilMate's primary design goals are usability, security, scalability, and maintainability. To deliver a smooth experience, the app will have a straightforward and user-friendly design, with complex algorithms hidden behind the UI layer. Sensitive data protection is of utmost importance; hence the app will be created with backend and frontend security in mind, employing powerful encryption mechanisms. Another important goal is scalability, as the app needs to be able to accommodate increasing user traffic without degrading functionality or user experience. Last but not least, maintainability is a top priority. The app's architecture was created to enable developers to respond to issues as they arise and keep the app's functionality and user experience up to par.

2. Current Software Architecture

There currently is no system specifically tailored for university students. There are accommodation-sharing websites. However, they are severely lacking both in quantity and quality. For instance, the website expat.com[1] has a total of four adverts showing rooms, which is not nearly enough for students. Another website, erasmusu.com[2], does not offer much flexibility in requirements. Moreover, it includes a very simple filtering system, and everything is on one page. So it does not include matching, advanced filtering, or any quality-of-life improving features

3. Proposed Software Architecture

3.1 Overview

The solution to solve the shared accommodation problem mentioned above will be provided through a social platform as a mobile application for the students to use. The app will provide

[1] https://www.expat.com/en/housing/middle-east/turkey/flat-share-house-share.html
[2] https://erasmusu.com/en/erasmus-istanbul/roommate

two options to newcomers: students looking for a housemate to share the rent and other costs with or those already looking for a house that other students are sharing.

At this point, both sides will start this process of finding housemates with their criteria. These criteria will reflect their priorities, such as the location of the house in terms of security, the proximity to the university or transportation lines, rent and other expenses, etc., as well as their concerns or preferences, such as pet allowance, smoking or drinking rules, loudness and socialness within the house, cleanliness, etc. All of these criteria and more will need to be filled in by the users. There is supposed to be a large number of criteria. Although only the crucial ones will be asked in the registration step of the application, the other less important ones, such as the preferred departments of housemates or the number of housemates, will be asked at intervals during the use of the application, for the convenience of the use of the application.

With the data retrieved through the questionnaire, the app will find matches between the housemate-seekers and house-seekers. It will be frequently pointed out in the application that with more data fields that the user fills, it will be more probable to find a good match between the two user types. Having found a good match that satisfies the users, they will be able to contact each other either through the messaging portal of the application or through their contact details, such as phone numbers or e-mail addresses, in case they want to share. It will also be possible with the app to browse through listings of houses or housemates rather than go for a matching process.

BilMate will bring sustained innovation into the market of house-sharing in Turkey in the form of a service that will transform the way students choose their roommates or find a proper place to live in. Rather than going through 10s or even 100s of people trying to find a suitable room and roommate combination, BilMate will optimize that process by allowing the user to decide for himself/herself what a suitable roommate should be.

For the initial implementation, we are planning on prioritizing the essential features that allow the user to list postings, create a posting, edit their profile, etc. During the later stages of development, we would start implementing and refining the matching algorithm to suit each user’s needs. The most substantial risk that we might face during the development stage is the accuracy and reliability of the matching algorithm. If the algorithm does not prove to be sufficient on its own, we will add extra manual filtering and sorting to accompany it to offer the user a complete experience. However, if the algorithm proved to be sufficient with respect to the criteria that were established, a manual filtering system may not be needed but may still be implemented for convenience purposes.

3.2 Subsystem Decomposition

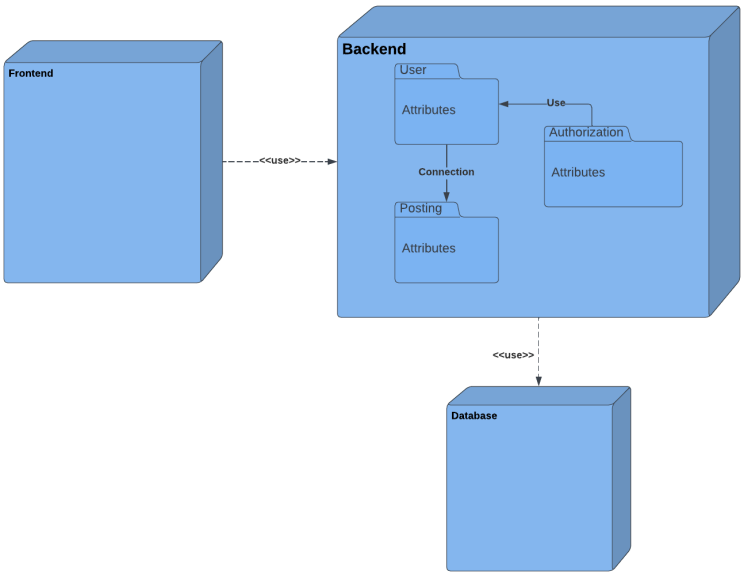

Figure 1: Subsystem Decomposition

Our system consists of three main subsystems; frontend, backend and database. Frontend is designed for mobile systems and communicates with backend through end points. Backend is planned to work in AWS server. Backend uses FastAPI and consists of three main packages, user, posting and authorization. User package deals with user accounts, authorization deals with authorization issues and security part of the system and posting package deals with all operations on posings. Database uses MongoDB and the backend directly communicates with the database through python libraries.

3.3 Access Control and Security

In the current iteration of BilMate, we have three types of users:

- Authenticated users: This encompasses all the users of BilMate
- Post Owners: This refers to the users who have created either a house-sharer or a house-seeker post on BilMate.
- Admin: This refers to the system admins responsible for managing the application and ensuring that the TOS are not violated. In case of a TOS violation, admins will take suitable action.

In the future, the BilMate team plans to allow limited, read-only access to BilMate for unauthenticated users. This means that unauthenticated users in the future will be able to see the posts but without the contact information of the respective posters. However, as of now, an unauthenticated user cannot access any part of the system.

In order to become an authenticated user, one must have a registered email address from Bilkent University. Security precautions have been taken to ensure that no bots or spammers will be in our ecosystem, such as spam detection and email verification.

After verifying the user's identity, they can create house-sharer and house-seeker posts that fit their needs to find the perfect roommate/flatmate.

Admins are the only people that have clearance to access the admin portal. The admin portal is used to enforce the TOS. In the event that a post or user is reported, the report will appear in the admin portal. Then one of the admins will decide the suitable action to be taken.

	Viewing/ Creating Posts	Updating/ Deleting Post	Reporting Posts/Users	Reviewing Reports	Banning Users
Unauthenticated User					
Authenticated User	**X**		**X**		
Post Owner	**X**	**X**	**X**		
Admin	**X**	**X**	**X**	**X**	**X**

Table 1: Access Matrix

4. Subsystem Services

4.1 User Management Service

The user management service is responsible for handling user-related operations such as registration, login, and account management. To implement this service, we will use Python Django, a high-level web framework that provides robust security features and easy integration with other services.

4.2 Post Management Service

The post management service is responsible for handling operations related to creating, updating, and deleting posts. To implement this service, we will use Python Django, which provides a rich set of features for building web applications. We will store posts' images in an Amazon S3 bucket for scalability and high availability.

4.3 Matching Service

The matching service is responsible for matching posts with users based on their preferences. To implement this service, we will use a machine learning algorithm, Boltzmann Machines, that will be trained on a large dataset of posts and user preferences.

4.4 Notification Service

The notification service is responsible for sending notifications to users when there are new posts or when their preferences change. We will use Django signals to trigger notifications when a new post is created or when a user's preferences change.

4.5 Search Service

The search service is responsible for providing search functionality to allow users to find posts based on keywords or other criteria.

In summary, the subsystem services include user management, post management, matching, notification, and search services. We will use a combination of Python Django and Amazon S3 to implement these services.

5. Test Cases

5.1 Functional Test Cases

5.1.1 Unit Testing

5.1.1.1 User Signs up with a non-Bilkent email

ID: 1

Test Type: Security

Procedure of Testing Steps:

- Enter a Bilkent email address during signup and complete the signup process to make sure everything works ideally
- Go back to signup page and now enter a non-Bilkent email address for signup
- Make sure that the signup process halts with a non-Bilkent email
- Make sure there is a user prompt for the reason it halts which is about non-Bilkent email.

Expected Result: The sign-up flow will fail, and the user will not gain access to BilMate
Priority: Critical

5.1.1.2 User tries to sign in without verifying their email

ID: 2

Test Type: Security

Testing Steps:

- Sign up with a valid Bilkent email address.
- Log out and try to sign in before verifying the email.
- Verify that an error message appears informing the user that they need to verify their email before logging in.

Expected Result: The login flow will fail, and the system will instruct the user to verify their email.

Priority: Critical

5.1.1.3 User tries to sign in with wrong email-password combination

ID: 3

Test Type: Security

- Sign up with a valid Bilkent email address and password.
- Attempt to log in using the correct email but an incorrect password.
- Verify that an error message appears indicating the user has entered an incorrect password.
- Attempt to log in using an email that is not associated with the account.
- Verify that an error message appears indicating that the user has entered an incorrect email or password.

Expected Result: The login flow will fail, and the user will be asked to write the correct email-password combination or use the “Forgot Password” option.

Priority: Critical

5.1.1.4 User tries to log in using the valid credentials

ID: 4

Test Type: Security

- Sign up with a valid Bilkent email address and password.
- Log out and attempt to log in using the correct email and password.
- Verify that the user is able to log in and access the application's main features.

Expected Result: The login follow will succeed granting said user access to BilMate.

Priority: Critical

5.1.1.5 User tries to post an incomplete post

ID: 5

Test Type: Functionality

- Log in to the application.

- Navigate to the post creation page and attempt to create a post without filling in all the required fields.
- Verify that an error message appears indicating that the user must fill in all required fields before submitting the post.
- Fill in all required fields and submit the post.
- Verify that the post appears correctly on the main feed.

Expected Result: The post verification system will not allow the post to go live, and the system will instruct the user to fill in the missing fields.

Priority: Major

5.1.1.6 User tries to post a complete posting

ID: 6

Test Type: Functionality

Procedure of Testing Steps:

- Log in to the application.
- Navigate to the post creation page and fill in all required fields.
- Submit the post.
- Verify that the post appears correctly on the main feed.

Expected Result: The post verification system will allow the post to go live

Priority: Minor

5.1.1.7 User tries to delete their own post

ID: 7

Test Type: Functionality

Procedure of Testing Steps:

- Log in to the application.
- Create a post.

- Click the delete button for the post created by the user.
- Verify that the post is successfully deleted and does not appear on the main feed anymore.

Expected Result: The identity management system will grant the user the privileges of deleting their own post

Priority: Major

5.1.1.8 User tries to delete another user’s post

ID: 8

Test Type: Security

Procedure of Testing Steps:

- Log in to the application.
- Create a post.
- Log out and create another user account.
- Attempt to delete the post created by the first user.
- Verify that an error message appears indicating that the user is not authorized to delete another user's post.

Expected Result: The identity management system will reject access to deletion flow for users that do not own the post.

Priority: Critical

5.1.1.9 User tries to update their own post

ID: 9

Test Type: Functionality

Procedure of Testing Steps:

- Log in to the application.
- Create a post.
- Click the edit button for the post created by the user.
- Change the content of the post.
- Submit the updated post.

- Verify that the post is updated and appears correctly on the main feed.

Expected Result: The identity management system will grant the user the privileges of updating his/her post.

Priority: Minor

5.1.1.10 User tries to update another user’s post

ID: 10

Test Type: Security

Procedure of Testing Steps:

- Log in to the application.
- Create a post.
- Log out and create another user account.
- Attempt to update the post created by the first user.
- Verify that an error message appears indicating that the user is not authorized to update another user's post.

Expected Result: The identity management system will reject access to (post) update flow for users that do not own the post

Priority: Critical

5.1.1.11 User tries to update their profile

ID: 11

Test Type: Functionality

Procedure of Testing Steps:

- Log in to the application.
- Navigate to the user profile page.
- Click the edit button for the user profile.
- Change the user profile information.

- Submit the updated profile.
- Verify that the user profile is updated correctly.

Expected Result: The identity management system will grant the user the privilege of modifying their account.

Priority: Minor

5.1.1.12 User tries to update another user’s profile

ID: 12

Test Type: Security

Procedure of Testing Steps:

- Log in to the application.
- Create a post.
- Log out and create another user account.
- Attempt to update the profile of the first user.
- Verify that an error message appears indicating that the user is not authorized to update another user's profile.

Expected Result: The identity management system will reject access to the (account) updating flow of another user

Priority: Critical

5.1.2 Sanity and Smoke Testing

5.1.2.1 Duplicate Users

ID: 13

Test Type: Security

Procedure of Testing Steps:

- Attempt to create two user accounts with the same email address.

- Verify that the system does not allow the creation of two user accounts with the same email address.

Expected Result: If a user already registered an account and they attempt to create another account using the same email. The registration system will reject the attempt.

Priority: Critical

5.1.2.2 Missing Fields During Registration

ID: 14

Test Type: Functional

Procedure of Testing Steps:

- User fills in incomplete information during registration
- User clicks the register button
- System verifies the information provided

System rejects the request made by the user

Expected Result: If, during registration, the user does not provide full information such as the email, password, major, etc. The registration system will reject the request made by that user

Priority: Critical

5.1.2.3 User inputs a wrong email

ID: 15

Test Type: Functional

Procedure of Testing Steps:

- User provides a wrong email address during registration
- User clicks the register button
- System verifies the email address
- System sends a verification code to the email address provided by the user
- User tries to verify their account using the wrong email address

Expected Result: In this case, the verification code will be sent to the wrong email, and the user will not be able to verify their account

Priority: Major

5.1.2.4 User tries to star a post that does not exist

ID: 16

Test Type: Functional

Procedure of Testing Steps:

- User attempts to star a post that has been soft deleted
- User clicks the star button on the post
- System verifies the isDeleted flag of the post
- Post should not appear in the starred list of any user

Expected Result: Due to the nature of soft removal, a post will remain in the database for a certain amount until it is actually deleted. In order to apply soft deletion, an isDeleted flag is set to true, and the post no longer appears in the results. So in the case that a post has been deleted, it should not appear in the starred list of any user

Priority: Minor

5.1.2.5 User requests a list based on their preference

ID: 17

Test Type: Functional

Procedure of Testing Steps:

- User logs in with valid credentials
- User selects the list preference option
- System retrieves list of postings personalized to the specific user
- System displays the personalized list of postings to the user

Expected Result: Assuming the user provides the valid credentials, a list of postings will be returned that is personalized to the specific user, rather than for other users

Priority: Critical

5.1.3 Regression Testing

5.1.3.1 When connection to the database is lost while processing a request

ID: 18

Test Type: Non-functional

Procedure of Testing Steps:

- Disconnect the database while a request is being processed
- Observe the system's behavior
- Check if the system attempts to recover
- Check if the system shuts down gracefully with suitable error messages if the recovery fails
- Verify that the system restarts gracefully
- Ensure that the system works without any side-effects

Expected Result: Due to the nature of Promises and asynchronous code, there are safety mechanisms to ensure in case of unexpected behavior (lost connection, large requests, long loading times, etc.). The system will attempt to recover. If that fails, the system will shut down and restart gracefully with suitable error messages.

Priority: Critical

5.1.3.2 Effect of post creation on the matching algorithm

ID: 19

Test Type: Functional

Procedure of Testing Steps:

- Create a post as user A
- Verify that the post does not appear on user A's recommendation page or anywhere else
- Check if the post appears only on user A's posts page
- Verify that the post does not appear on any other user's page
- Ensure that the matching algorithm works correctly after the post creation

Expected Result: When a user creates a post, let’s call him user A. That post should not appear on user A’s recommendation page or anywhere else, even if they actively search for it. The only location it should appear in is A’s posts page.

Priority: Major

5.1.3.3 Effects of starring a post on the matching algorithm

ID: 20

Test Type: Functional

Procedure of Testing Steps:

- Star a specific post as a user
- Verify that the algorithm re-calibrates the parameters
- Check if the system serves the user better postings in the future
- Ensure that the system works without any side-effects

Expected Result: When a user stars a specific post, the algorithm will re-calibrate the parameters to serve the user better postings in the future.

Priority: Major

5.1.3.4 Privacy Settings

ID: 21

Test Type: Non-functional

Procedure of Testing Steps:

- Change the account's privacy settings to high privacy
- Verify that all posts that were made (active and inactive) have the phone number hidden from all the other users of the system
- Check if the system works without any side-effects

Expected Result: When a user decides to change their account’s privacy settings to high privacy, all posts that were made (active and inactive) will have the phone number hidden from all the other users of the system.

Priority: Minor

5.1.3.5 User does not indicate preferences

ID: 22

Test Type: Functional

Procedure of Testing Steps:

- Create a user without indicating any preferences
- Verify that the system assigns all parameters to be equally important
- Ensure that the system matches users accordingly
- Check if the system works without any side-effects

Expected Result: In this case, the system will assign all parameters to be equally important and match the users accordingly; this will change when either the user manually changes the preferences or through extended use of BilMate’s starring system, which includes a feedback loop to improve the algorithm.

Priority: Minor

5.1.4 System Testing

5.1.4.1 User signs up

ID: 23

Test Type: Component

Procedure of Testing Steps:

- User fills out registration form with valid information and submits it.
- System should verify the email address belongs to a Bilkent University student.
- Verification code should be sent to the user's email.
- User inputs verification code and is directed to their profile.
- User should have no preferences or posts initially.

Expected result: The system should register the user, then after verifying that the email belongs to a Bilkent University student, a verification code will be sent. After verification, the user should be able to access the system with initially no preferences and no posts, until they create a post.

Priority: Critical

5.1.4.2 User scrolls through posts

ID: 24

Test Type: Functional

Procedure of Testing Steps:

- User logs in to the system with valid credentials.
- User scrolls through the posts.
- User selects a post and views further details.
- User contacts the owner of the post.

Expected Result: After a user has been authenticated, they can scroll through BilMate's selection of posts; the user can then press on any post and will be redirected to the respective post, which he/she will be able to view further details and contact the owner of the post.

Priority: Major

5.1.4.3 User deletes a post

ID: 25

Test Type: Component

Procedure of Testing Steps:

- User logs in to the system with valid credentials.
- User heads to the manage posts section.
- User selects a post to delete.
- User confirms the deletion of the post.
- The post is initially removed from the system with soft removal, followed by hard removal.

Expected Result: A user needs to be authenticated to carry out this action. The post will be initially deleted from the system with soft removal, followed by hard removal. At the point of soft removal, the post will stop appearing for any other users of BilMate.

Priority: Minor

5.1.4.4 Reporting a user

ID: 26

Test Type: Functional

Procedure of Testing Steps:

- User logs in to the system with valid credentials.
- User navigates to the report section.
- User selects a post or user to report.
- User submits the report.
- Admins review the report ticket.
- Admin approves or rejects the report ticket.

Expected Result: If a user of BilMate believes that a post or another user is breaking the ToS of BilMate by uploading content deemed inappropriate, they can report the post or the user to the admins. After a review process, the admin can approve the ticket, thus deleting the post. Or it could be rejected, which results in the post remaining on the platform.

Priority: Major

5.1.4.5 Account deletion

ID: 27

Test Type: Functional

Procedure of Testing Steps:

- User logs in to the system with valid credentials.
- User navigates to the account deletion section.
- User inputs a valid password for account verification.
- User confirms the account deletion.
- All posts and account information should be deleted from the system.

Expected Result: In case a user no longer wants to use BilMate, they have the right to be forgotten. After confirmation, all user posts, in addition to their account, will be removed from the system.

Priority: Critical

5.1.5 Acceptance Testing

5.1.5.1 Can users register for BilMate

ID: 28

Test Type: Functional

Procedure of Testing steps:

- Navigate to BilMate's registration page
- Fill in all the required fields with valid information
- Submit the registration form
- Check if a confirmation message is displayed on the screen
- Verify that a verification email is sent to the provided email address

Expected Result: The users who test BilMate should be able to seamlessly create an account, assuming they have an active Bilkent email

Priority: Major

5.1.5.2 Can users verify their email

ID: 29

Test Type: Functional

Procedure of Testing steps:

- Navigate to the BilMate email verification page
- Click on the verification link in the email sent to the user
- Enter the verification code provided in the email
- Click on the verification button
- Verify that the email is verified successfully

Expected Result: The users should be able to access their email provider and verify their email

Priority: Major

5.1.5.3 Can users create a post

ID: 30

Test Type: Functional

Procedure of Testing steps:

- Navigate to the BilMate post creation page
- Fill in all the required fields with valid information
- Submit the post creation form
- Check if a confirmation message is displayed on the screen
- Verify that the created post is visible on the BilMate home page

Expected Result: The users should be able to fill in a form and create a post, assuming all the fields are valid

Priority: Major

5.1.5.4 Can the users update their post

ID: 31

Test Type: Functional

Procedure of Testing steps:

- Navigate to the BilMate post update page
- Select a post to update
- Modify any of the fields in the post
- Submit the post update form
- Check if a confirmation message is displayed on the screen
- Verify that the updated post is visible on the BilMate home page

Expected Result: After creating a post, the user should be able to update fields such as date, location, price, etc. on their post

Priority: Minor

5.1.5.5 Can the user delete their post

ID: 32

Test Type: Functional

Procedure of Testing steps:

- Navigate to the BilMate post deletion page
- Select a post to delete
- Confirm the deletion
- Check if a confirmation message is displayed on the screen

- Verify that the deleted post is no longer visible on the BilMate home page

Expected Result: After creating a post, the user should be able to delete that post from the system

Priority: Major

5.1.5.6 Can the user star any post

ID: 33

Test Type: Functional

Procedure of Testing steps:

- Navigate to the BilMate post page
- Select a post to star
- Click on the star button
- Verify that the post is starred successfully

Expected Result: The user should be able to star any post he/she finds interesting, however, the only limitation is that the user should not be able to star their own posts.

Priority: Minor

5.1.5.7 Can the user update their own profile

ID: 34

Test Type: Functional

Procedure of Testing steps:

- Login to BilMate
- Navigate to the profile management section
- Update several fields such as password, number, preferences, etc.
- Save the changes

Expected Result: The user should be able to update several fields, such as password, number, preferences, etc., in the profile management section of BilMate.

Priority: Major

5.1.5.8 Can the user delete a post

ID: 35

Test Type: Functional

Procedure of Testing steps:

- Login to BilMate
- Navigate to the post that the user wants to delete
- Click on the delete button
- Confirm the deletion

Expected Result: The user, assuming they are authenticated, should be able to delete their account from the system.

Priority: Major

5.1.5.9 Can the user send another verification email

ID: 36

Test Type: Functional

Procedure of Testing steps:

- Login to BilMate
- Navigate to the verification email section
- Request another verification code to be sent to their email
- Check the email for the new verification code

Expected Result: If the verification email expired, the user should be able to request another verification code to be sent to their email.

Priority: Minor

5.1.5.10 Can the user upload images to their posts

ID: 37

Test Type: Component

Procedure of Testing steps:

- Login to BilMate
- Navigate to thecreate post section
- Fill in the post details
- Upload picture(s)

Expected Result: The newly created post should have the images displayed.

Priority: Critical

5.1.5.11 Can the user report a post

ID: 38

Test Type: Functional

Procedure of Testing steps:

- Login to BilMate
- Navigate to the view postings section
- Report a user by including a reason and description of the offense

Expected Result: The reporting request will be sent and allocated to an admin for review.

Priority: Major

5.1.5.12 Can an admin review and accept a reporting ticket

ID: 39

Test Type: Functional

Procedure of Testing steps:

- Login to BilMate’s admin page
- Navigate to the review reports section
- Press on a report request and view the details
- Accept the report request, thus approving that it is a valid offense

Expected Result: The offending post will be deleted, and the owner of that post will be warned.

Priority: Major

5.1.5.13 Can an admin review and reject a reporting ticket

ID: 40

Test Type: Functional

Procedure of Testing steps:

- Login to BilMate's admin page
- Navigate to the review reports section
- Press on a report request and view the details
- Reject the report request, thus approving that the post does not break the ToS

Expected Result: The offending post will remain active on BilMate.

Priority: Major

5.1.6 Input validation

5.1.6.1 Registration

ID: 41

Test Type: Functional

Procedure of Testing steps:

- Navigate to the registration page
- Attempt to submit a registration form with invalid inputs for each of the required fields (email, phone number, nationality, university major)
- Verify that an appropriate error message is displayed for each invalid field
- Submit the form with valid inputs for all required fields
- Verify that the user is successfully registered and redirected to the login page

Expected Result: Several fields during registration need to be validated. As previously mentioned, all the emails must end with bilkent.edu.tr. Additionally, a phone number must start with the international code (+90 in the case of Turkey) and be of a specific length. Nationality and university majors are already limited to a specific list of enumerables; this ensures consistency among all users.

Priority: Major

5.1.6.2 Updating Fields

ID: 42

Test Type: Functional

Procedure of Testing steps:

- Log in to BilMate with valid credentials
- Navigate to the profile management section

- Attempt to update each of the optional fields (password, preferences, importance)
- Verify that an appropriate error message is displayed for any invalid input
- Submit the form with valid inputs for all optional fields
- Verify that the updated fields have been successfully saved to the user's profile

Expected Result: While updating, there are optional fields that they could leave as-is, or update them to improve their experience better. For instance, they can update their password, which will need to be confirmed. Moreover, they have the option to update their preferences and the importance of those parameters. The importance will be validated against a list of choices, with the same process applying to the preferences themselves.

Priority: Medium

5.2 Non-Functional Test Cases

5.2.1 Performance Testing

ID: 43

Test Type: Performance Testing

Procedure of Testing steps:

- Set up a sizable database with a certain amount of data.
- Send queries to the backend with increasing complexity and data size.
- Measure the response time for each query.
- If the response time is above 500 ms, investigate and optimize the query or the database performance.
- Once the data size reaches a certain upper threshold, ensure that the response time is at most 1000 ms.

Expected Result: The backend should respond to a query within 500 ms at most with a sizable database size. If the stored data reaches a certain upper threshold, the response time should be 1000 ms at most.

Priority: Critical

5.2.2 Usability Testing

ID: 44

Test Type: Usability Testing

Procedure of Testing steps:

- Recruit human users to test the application.
- Observe how they use the application and take note of any difficulties or confusion they experience.
- Ask users to perform specific tasks and monitor their progress.
- Evaluate the ease of use, logical transitions between pages, and the quality of error messages.
- Assess the consistency and informativeness of the GUI.

Expected Result: The usability of the application will be evaluated by human users. They should be comfortable using the application. To ensure this, the application should be easy-to-use, have logical transitions between pages, and give meaningful error messages. The GUI of the application should be informative and consistent.

Priority: Major

5.2.3 Stress Testing

5.2.3.1 High Load

ID: 45

Test Type: Stress Testing

Procedure of Testing steps:

- Simulate simultaneous access by all Bilkent users to BilMate
- Monitor server uptime using AWS tools
- Verify that table partitioning and load balancing have ensured the server stays up

Expected Result: Assuming all Bilkent users access BilMate simultaneously, table partitioning and load balancing offered by AWS will ensure that our server stays up. This

is a worst-case scenario; a more realistic scenario will not have any issues regarding server uptime.

Priority: Critical

5.2.3.2 High Stress

ID: 46

Test Type: Stress Testing

Procedure of Testing steps:

- Use bots to flood the server
- Verify that the spam detection system catches the behavior and rejects further attempts to create/delete posts until a timer expires

Expected Result: The spam detection system should be effective in catching bot behavior and preventing server overload. If a user uses a bot to flood the server, our spam detection system will catch the behavior and reject further attempts to create/delete posts until a timer expires.

Priority: Major

5.2.4 Security Testing

ID: 47

Test Type: Security Testing

Procedure of Testing steps:

- Attempt to gain unauthorized access to BilMate's GitHub repository, MongoDB Atlas instance, or AWS EC2 server using various methods, such as brute force attacks, injection attacks, and cross-site scripting attacks.
- Monitor whether the respective service alerts all team members and automatically secures the login portal to stop any further attempts of breaking in.
- Verify whether the system uses JWT authentication tokens to ensure that only authorized users can perform protected actions, such as deleting or updating personal information.

Expected Result: In case an adversary attempts to gain unauthorized access to either BilMate's GitHub repository, MongoDB Atlas instance, or AWS EC2 server, the respective service will alert all team members and automatically secure the login portal to stop any further attempts of breaking in. The system also uses JWT authentication tokens to ensure that only authorized users can perform several protected actions, such as deleting or updating personal information.

Priority: Critical

5.2.5 Portability Testing

ID: 48

Test Type: Portability

Testing Procedure of Testing steps:

- Install BilMate on Android and IOS platforms.
- Use the application on both platforms to verify its functionalities.
- Check if the application works in the same way on both platforms and doesn't give platform-dependent errors.

Expected Result: Users may have different operating systems, Android or IOS. The application should work in the same way and should not give platform-dependent errors. The functionalities of the application should not be affected by the platform in which it is used.

Priority: Major

5.2.6 Maintenance Testing

ID: 49

Test Type: Maintenance

Testing Procedure of Testing steps:

- Add new functionalities and modify existing ones in the application
- Run maintenance testing on the altered and added features

- Check that the altered and added features work as expected and do not cause any side-effects for the unaltered features

Expected Result: After the launch of the application, it will require several updates for adding new functionalities and altering the existing ones. Those updates are supposed not to affect the work of unaltered features and cause any side effects. Maintenance testing ensures that the altered and added features work as expected and do not cause any side effects for the other parts of the system.

Priority: Major

5.2.6.1 Confirmation Testing

ID: 50

Test Type: Confirmation

Testing Procedure of Testing steps:

- Modify functionalities and features in the application
- Run confirmation testing on the modified functionalities and features
- Check that the modified functionalities and features work properly and expectedly

Expected Result: Confirmation testing tests the modified functionalities and features to ensure they work properly and expectedly.

Priority: Major

5.2.6.2 RegressionTesting

ID: 51

Test Type: Regression

Testing Procedure of Testing steps:

- Modify functionalities and features in the application
- Run regression testing on the remaining system with unmodified functionalities and features
- Check that the modifications do not affect the running of the system and do not cause any side-effects

Expected Result: Regression testing tests the remaining system with unmodified functionalities and features to ensure modifications do not affect the running of the system and do not cause any side-effects.

Priority: Critical

5.2.7 Recovery Testing

ID: 52

Test Type: Recovery

Testing Procedure of Testing steps:

- Simulate different failure types, including power interruption, network failures, database overload, and hardware failures
- Check that the system terminates gracefully without data corruption in case of failure

Expected Result: In the case of failure, the system should terminate gracefully without data corruption. Recovery testing contains simulations of different failure types, including power interruption, network failures, database overload, hardware failures…

Priority: Critical

5.2.8 Reliability Testing

ID: 53

Test Type: Reliability

Testing Procedure of Testing steps:

- Test the system in a particular environment for a fixed period of time
- Ensure that the system works properly without any failure in that environment
- Ensure that the system yields the same output in the same setup to ensure its reliability
- Test the system according to its maximum workload, the proper execution of each operation and feature, and the working of an entire system after a bug fix

Expected Result: The system should work properly without any failure in a particular environment for a fixed period of time, and it should yield the same output in the same setup to ensure its reliability. The system should be tested according to its maximum workload, the proper execution of each operation and feature, and the working of an entire system after a bug fix.

Priority: Major

6. Consideration of Various Factors in Engineering Design

This section outlines some factors that might impact the project's development.

6.1 Public Health

Issue of public health does not affect the project. We expect users' own control in case of a health-related situation. A house-sharer and house-seeker must plan according to their current health situation.

6.2 Public Safety

To ensure the safety of users, a reporting feature will be added to the project. In case of encountering a malicious post, a user can report the post with a comment explaining the reasoning behind the report. This request will be taken and sent to a list of reported posts, where each post contains a counter displaying the number of times that post is reported. An admin of the application will observe each post and corresponding comments, and if feasible, the reported user and their posts will be deleted and will be banned for an indefinite duration.

Also, as part of the project's terms, no user data will be publicly shared or with an untrusted third-party app.

6.3 Public Welfare

Primary aim of the project is to connect students willing to share their house with students willing to stay in a student house to reduce their costs and enjoy a better accommodation environment. Besides this user life improvement, to enhance the app's usability, the application's lightness, and a fluent interface will be a primary design concern. Thus, comfort for users will be provided both on the application side and in real life.

6.4 Global

As mentioned in the previous subsection, the project's target audience is university students who own a house and/or are willing to stay in one of those houses. Most of the universities in Turkey teach in English, but to extend the scope and ease the use for those who do not know much English, in addition to the application’s primary language, which is English, Turkish might be added in later stages of development.

6.5 Cultural

For Turkish and many other international students, staying in a student house is an accepted habit; thus, we do not expect many issues from the cultural side. However, further preference options could be added to the preferences list, such as users’ religion, diet, etc., to enhance the output of the matching algorithm.

6.6 Social

The project expects a user to provide their email and phone number, which will be used to connect a house-seeker and sharer. Unless specified, both contact information of a user will be displayed publicly, but a user can hide their phone number, which is more prone to malicious acts. If they do, their email addresses will be the only way to connect a house-sharer and seeker. This way, even though there might be a little drop in UX, users’ security will be provided.

6.7 Environmental

Since some students will be allocated to houses rather than dormitories, this will enhance the capacity of the student dorms. This way, those who are in actual need can stay in dorms, and others can stay with some house-sharer by increasing their accommodation comfort while reducing their bills.

6.8 Economic

BilMate will be a free-to-use application since its launch. However, donation methods, such as “buy me a coffee” might be added to provide a way of income to developers.

	Effect Level	Effect
Public Health	0	We expect users' own intervention in case of a public health situation
Public Safety	8	Forbiddance of users with malicious intents, Protection of users' data
Public Welfare	7	Lightweight application, Outstanding User Interface along with User Experience
Global	2	Extend the language of the project
Cultural	1	Adding extra required preferences to the preference list
Social	8	Hiding a user's phone number, Providing a way of communication in case of hidden user phone number
Environmental	0	Scope of the environment is not of part of the project
Economic	1	Add donation section for a way of income

Table 2: Factors that can affect analysis and design

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

As a team, we managed to construct a collaborative working environment. Everyone in our group took on a responsibility and acted accordingly. Group meeting sessions were held when required, and further discussions regarding new responsibilities and work division were made. During the meeting sessions, each person was required to voice his/her opinion regarding decisions such as adding new features or removing

out-of-scope functionalities in addition to the software architecture approach. This ensured that everyone contributed equally to the success of our project.

Ahmet, Ebrar, and Atasagun are part of the team responsible for the development of the project's back-end, while Javid and Onuralp were responsible for the front-end section of the project. Ahmet's primary contribution was designing the backend's workflow, setting up AWS for future hosting, creating the database schema, and preparing interfaces for front developers to access data. Ebrar's primary responsibility was to develop the security (authentication and authorization) section of the APIs, namely, providing JSON Web Tokens (JWT) along with some other enhancements regarding the back-end's security. Meanwhile, Atasagun's responsibilities were to provide user interface design of the project, and logo design, ensuring fast response times when retrieving and storing images in AWS S3, and creating the admin portal and the reporting system.

Per the provided UI design of Atasagun, Javid and Onuralp divided the project pages and assigned their sections accordingly. Due to Onuralp's experience in Android development, he always shed light for Javid, a novice in this field. Both, in coordination and communication, developed many pages of the project in accordance with the object-oriented software development principles, such as classes, objects, inheritance, etc. Note that the front-end team used third-party libraries to enhance the functionalities of the front-end section.

7.2 Helping creating a collaborative and inclusive environment

As mentioned in the previous subsection, we managed to construct a collaborative environment where each individual was assigned a task, and with it comes responsibility. In addition, an inclusive environment, where each team member can freely express their ideas regarding any aspect of the project, was ensured from the beginning. All group members being in the same department helped a lot in the process of this environment's creation. Additionally, the majority of the members' connection from the past also contributed to this construction.

Ahmet, the primary organizer of the group, mostly helped in terms of the establishment of the group's internal systems. Onuralp, for the most part, acted as a

mentor in the front-end section's development process for Javid, taking all queries from him seriously. Other members contributed much to ensure that the group has a friendly tone.

All in all, our group is a collaborative and inclusive environment where all individuals' ideas and opinions are taken seriously and welcomed, regardless of their gender, religion, race, or any other aspects of their life. In this project, everyone is equal and shares the same responsibility. The success of the project is attributed to everyone on the team.

7.3 Taking lead role and sharing leadership on the team

Ever since the beginning of the project, we agreed that we wouldn't have a leader or leaders in the traditional sense. As we are all adults with full autonomy and ideas that will contribute to the project's success, we have decided to take a more flexible approach to manage the work division and the general organization of the project.
The team divided the 5-member team into two sub-teams, one responsible for the back-end development (API, Database, AWS integration) and the other responsible for the front-end development. In order to avoid micromanagement[3], an unofficial leader for each group was chosen by consensus and their merit in their respective fields. But it is important to reiterate that all the team members are equal and have to contribute equally to the project's success.
For instance, in the front-end team, Onuralp, being the one with the most experience in Android development, carried out the tasks of a team lead by choosing the technology and setting up the project according to object-oriented practices. It follows that Onuralp was mainly concerned with setting up a strong foundation for the project. Meanwhile, Javid was crucial in managing communication between the front-end and back-end teams. This role is crucial as a miscommunication error could result in severe delays and possible financial losses.
On the other side, in the back-end team, each member had a specialty. As such, depending on the topic at hand, one person's opinion had more merit as they had more

[3] https://www.investopedia.com/terms/m/micro-manager.asp

experience. For example, Atasagun specializes in UI design and ensuring a smooth UX experience, so he was tasked with designing the UI, and we all agreed that he was the fittest out of all the team members. While Ebrar had years of experience designing secure APIs and web systems, it was in the project's best interest to delegate the job of the app's security to Ebrar as she was the most competent. Finally, Ahmet had some experience in developing full-stack applications and database design, as well as experience with AWS services. As such, we agreed to assign the job of setting up the project, designing the database, and integrating it with AWS to him.

8. References

[1] What Is Software Testing. [Online]. Available: https://www.softwaretestinghelp.com/manual-testing-tutorial-1/. [Accessed: 08 March 2023]